Personalización de comportamiento en ambientes inteligentes empleando una herramienta para superficies interactivas

Patricia Pons Tomás

Trabajo Final de Máster

Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

Departamento de Sistemas Informáticos y Computación

Dirigido por:

Dr. Francisco Javier Jaén Martínez

Dr. Alejandro Catalá Bolós

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Septiembre 2013, Valencia

Resumen

Los espacios inteligentes se están convirtiendo en un futuro prometedor que aportarán indudables mejoras a la sociedad a medida que el número de dispositivos inteligentes capaces de formar parte de dichos entornos crece rápidamente. Puesto que la combinación de tantos y tan variados elementos inteligentes en un mismo espacio abre la puerta a nuevas oportunidades para mejorar la calidad de vida de las personas, también hace más complejo el desarrollo de sistemas capaces de adaptarse a la perfección a cualquier contexto o usuario. Y más todavía si se tiene en cuenta la naturaleza cambiante de los entornos o de las preferencias de las personas que los habitan. Por tanto, la personalización del comportamiento de estos sistemas inteligentes por parte de los usuarios finales de los mismos se plantea como una tarea de sumo interés. Sin embargo, puesto que estos usuarios no son expertos en el sistema ni suelen tener conocimientos de programación, es necesario proveer mecanismos sencillos para llevar a cabo una tarea compleja como es la especificación de comportamiento.

Este trabajo presenta un editor visual basado en expresiones de flujos de datos para superficies interactivas que permitirá definir reglas de comportamiento en los futuros espacios inteligentes de manera natural y sencilla. Se ha llevado a cabo un estudio para demostrar la usabilidad del editor en base a la capacidad de usuarios no programadores de comprender las abstracciones y conceptos en los que se basa la herramienta, la facilidad de uso de la interfaz visual propuesta y la adecuación de los mecanismos de interacción multitacto y tangibles utilizados en la edición. Los resultados de este trabajo han demostrado que usuarios sin conocimientos previos de programación son capaces de utilizar correctamente la herramienta propuesta para definir reglas de comportamiento de cierta complejidad en el contexto de una casa inteligente.

El estudio realizado ha permitido, además, detectar problemas leves de usabilidad a los que se ofrecerá solución en próximas versiones de la herramienta, y han surgido diversas líneas de actuación para acercar y hacer todavía más natural la personalización de ambientes reactivos.

Palabras clave

Reglas, personalización, definición de comportamiento, eventos, espacios inteligentes, lenguaje visual, superficies interactivas, inteligencia ambiental, interfaces tangibles de usuario.

Agradecimientos

A mis padres, por el cariño, la fuerza y el apoyo que me transmiten cada día.

A mi familia, que siempre está ahí a cada paso que doy.

A Pedro, por haberme regalado seis años inolvidables.

A Javier y Alejandro, para quienes no tengo suficientes palabras de admiración y agradecimiento. Su dedicación, guía y confianza me han hecho crecer en lo personal y en lo profesional.

A Fernando y Vicente, junto a quienes he recorrido este camino, por el apoyo mutuo y por haber hecho cada día más divertido.

A mis compañeros de laboratorio, por las risas, los buenos momentos, y los sabios consejos.

A mis compañeros de facultad, en especial a Carol, María, Joan, Carles, Andreu, Juan, Javi, Pablo, Jorge y Juanma, porque no podría haber encontrado personas más grandes ni amigos más incondicionales.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Introducción a la Inteligencia Ambiental

La Inteligencia Ambiental, o *Ambient Intelligence* (AmI) *(Cook et al., 2009)(Shadbolt, 2003)* es un campo joven de la informática, cuyo objetivo principal consiste en dotar de inteligencia a los elementos de cualquier entorno, integrándolos de manera natural y transparente *(Norman, 1998)*, de modo que ayuden o complementen las tareas de los usuarios de dicho entorno. Por tanto, es un área de investigación inherentemente centrada en el usuario, en las necesidades del mismo y en ampliar sus capacidades, así como en las potenciales mejoras que se podrían aportar en la calidad de vida de las personas en distintos entornos: en casa, en la oficina, en el coche, en hospitales, etc.

Dentro de la Inteligencia Ambiental se engloban numerosas disciplinas, a destacar, entre otras, las siguientes: computación ubicua *(Weiser, 1991)*, donde se pretende que los elementos de computación sean lo más transparentes posibles al usuario; sensibilidad al contexto, según la cual el entorno debe ser capaz de adaptarse a los cambios que se producen dentro del mismo y responder en consecuencia con la situación actual del usuario; monitorización y captación de señales, a fin de ser conscientes de lo que sucede dentro del entorno inteligente; o Interacción Hombre-Máquina, conocida como *Human-Computer Interaction* o HCI, que trata de facilitar y hacer más natural la comunicación entre los sistemas y los elementos de computación. Como puede observarse, todas estas disciplinas están interrelacionadas entre sí, ya que por ejemplo una interacción más natural con los sistemas, como la basada en gestos o voz, permitiría que los computadores fueran más transparentes al usuario, obteniendo sistemas más ubicuos. O la simbiosis entre sensibilidad al contexto y monitorización, ya que sin ser capaces de captar lo que sucede en el entorno, los sistemas inteligentes no podrían adaptarse al contexto del usuario. La mayor parte de los avances recientes en el campo de AmI están orientados a mejorar aspectos relacionados con estas disciplinas.

Otro de los aspectos sobre el que se está incidiendo últimamente en el área de AmI es la integración transparente y automática de la gran cantidad de elementos diversos que pueden formar parte de un entorno inteligente *(Gámez & Fuentes, 2011)(Uribarren et al., 2008)*. Cada dispositivo susceptible de ser utilizado en un entorno AmI puede provenir de un fabricante distinto, tener unas especificaciones determinadas, ofrecer una serie de servicios y además que dichos servicios deban ser invocados utilizando un *framework* específico o una tecnología concreta. Además, los entornos inteligentes están caracterizados por ser entornos dinámicos, es decir, que evolucionan con el tiempo, de modo que pueden incorporarse al entorno nuevos dispositivos mientras que otros quedarán obsoletos, o incluso algunos dispositivos pueden resultar inoperativos temporalmente debido a fallos. Por tanto, es necesario asegurar la compatibilidad, comunicación e integración de los mismos, sin que el usuario se vea involucrado en tareas que requieran configuración a bajo nivel.

1.2. Motivación

Si bien se ha comentado que la Inteligencia Ambiental se trata de una disciplina centrada en el usuario en aras de obtener sistemas ubicuos y transparentes, la mayoría de trabajos en este campo han evitado en todo lo posible la interacción explícita de los mismos con el sistema, de modo que la sensibilidad al contexto se da por medio de interacciones implícitas *(Schmidt, 2000)*, es decir, acciones que las personas realizan sin pretender comunicarse con el sistema, pero que son reconocidas como eventos ante los que debe reaccionar y adaptarse. Por tanto, la adecuación al contexto y la inteligencia del sistema recaen sobre la efectividad de tareas como el reconocimiento automático de las actividades del usuario, así como de los algoritmos de aprendizaje utilizados para la toma de decisiones. Sin embargo, existen una gran variedad de entornos y escenarios posibles donde un sistema inteligente puede desplegarse, y cada escenario conlleva infinidad de usuarios potenciales, donde cada usuario tendrá sus propias preferencias, requisitos o necesidades. Es más, las necesidades de una única persona pueden cambiar a medida que pasa el tiempo. En consecuencia, es improbable que ni desarrolladores ni técnicas basadas en aprendizaje automático lleguen a alcanzar niveles de precisión altos en todos los posibles casos de uso del sistema. Por ello, los sistemas inteligentes que se desarrollan se centran en un contexto específico de uso, sobre el que perfeccionar las técnicas de interacción, aprendizaje y respuesta. Esto da lugar a numerosos sistemas y líneas de investigación, que unido a sistemas comerciales y dispositivos propietarios, resulta en la nula adaptación de estos sistemas al día a día de los usuarios finales. Por tanto, parece que todavía queda mucho camino por andar para ver nuestro entorno diario como un ecosistema inteligente.

Al analizar otras áreas de la computación, puede observarse que en numerosos casos la información que proveen los usuarios de manera explícita resulta de alto valor para el desarrollo del sistema *(Churchill, 2013)(Teruel et al., 2012)*. En vista de esto, se ha comenzado a plantear la aplicación de estos mismos principios al área de AmI *(Holloway & Julien, 2010)*, de modo que los usuarios finales sean quienes, de un modo natural y no intrusivo, adapten o personalicen el sistema inteligente para que éste se comporte como ellos desean exactamente que lo haga. La interacción implícita resulta indiscutiblemente necesaria, pero complementar la misma con la interacción explícita de los usuarios supone un valor añadido que permitiría acercar el sistema inteligente a los mismos, de modo que lo sientan como más personal y no como un conjunto de dispositivos que reaccionan ante ellos sin saber muy bien por qué. Además, en vistas de la creciente complejidad de los futuros sistemas inteligentes y de la dificultad que tendrá en dichos escenarios adaptar el sistema automáticamente a diferentes contextos, se hace necesario involucrar a los usuarios finales en la personalización del comportamiento de estos entornos inteligentes. Tomar esta decisión incrementaría la comodidad con el entorno ya que este se ajustaría a las necesidades y gustos de cada persona con altísima precisión.

No obstante, tal tarea conlleva un gran desafío, ya que los usuarios finales de estos sistemas, que son quienes deben llevar a cabo la personalización de los

mismos, normalmente no tienen conocimientos de programación. Por tanto es importante buscar mecanismos que permitan llevar a cabo la personalización del comportamiento de manera natural para cualquier tipo de usuario, sin recaer en sistemas excesivamente sencillos que no tengan aplicación real. En la búsqueda de este tipo de soluciones, hay dos cuestiones a analizar: cómo va a ser expresado el comportamiento, es decir, qué tipo de construcciones o acciones se van a permitir, y por otro lado, qué interfaz va a permitir al usuario definir las reglas de comportamiento necesarias de manera natural e intuitiva.

En cuanto a cómo expresar el comportamiento de un sistema inteligente, dado que estos entornos son de naturaleza reactiva, la mayoría de soluciones adoptadas se basan en la respuesta ante eventos. Concretamente, las reglas tipo Evento-Condición-Acción o ECA son el mecanismo preferido para llevar a cabo estas especificaciones *(López de Ipiña, 2001)*. Además, hay diversos estudios que demuestran que este tipo de reglas ECA son fácilmente comprensibles por usuarios sin ningún tipo de experiencia en programación *(Good et al., 2010)(Kelleher & Pausch, 2005)(Pane et al., 2001)*. El problema consiste en que, hasta la fecha, los enfoques basados en reglas ECA han restringido la expresividad de las mismas a fin de hacer el sistema comprensible para usuarios finales, o bien la expresividad es tan alta que únicamente los desarrolladores profesionales han podido utilizarlos. Además, los escenarios en los cuales se han empleado reglas ECA utilizan ontologías específicas para el dominio en cuestión, restringiendo la especificación a un pequeño conjunto no extensible de elementos, y las evaluaciones en cuanto a la comprensión de estos modelos de reglas no involucran escenarios más reales y variados. En vista de las limitaciones de los sistemas existentes, se propone un lenguaje visual altamente expresivo e independiente del dominio para la definición de reglas de comportamiento en entornos reactivos. Se han realizado diversos estudios *(Catalá, Pons, et al., 2013)(Pons et al., 2011)* que revelan que este lenguaje es comprensible por usuarios sin conocimientos previos de programación.

La segunda problemática a abordar, en la cual se enmarca el presente trabajo, consiste en obtener una herramienta usable y adecuada, basada en el sistema de reglas propuesto, que permita a los usuarios finales de un sistema inteligente llevar a cabo de la manera más natural posible la especificación de comportamiento. En el campo de HCI han surgido nuevas plataformas y técnicas de interacción durante los últimos años que pretenden facilitar la comunicación con los ordenadores a través de interfaces más naturales. En concreto, las Interfaces Tangibles de Usuario (*Tangible User Interfaces* o TUIs) *(Ishii & Ullmer, 1997)* se basan en la idea de que las personas somos capaces de comprender el funcionamiento de los objetos físicos atendiendo a su forma y gracias a la manipulación directa de los mismos con nuestras manos. En cambio, la mera representación virtual de un objeto no permite que las personas seamos capaces de usar estas capacidades cognitivas para entender los objetos virtuales del mismo modo. Por ello, las TUIs emplean elementos tangibles ligados a representaciones visuales de los objetos, de modo que la entrada y salida del sistema no se encuentra tan diferenciada como en los sistemas convencionales, sino que todos los elementos se integran coherentemente y los objetos digitales responden ante cambios en los objetos tangibles, y viceversa. De este modo, la interacción con

estas plataformas se realiza generalmente empleando la manipulación directa con los dedos o con objetos físicos, empleando metáforas para determinadas acciones y consiguiendo interacciones extremadamente intuitivas. A la vista de las ventajas que ofrecen este tipo de interfaces, se ha decidido explorar el desarrollo de un editor de reglas para una TUI particular, como es el caso de una mesa interactiva, que ofrezca mecanismos de interacción natural mediante los dedos y objetos físicos, y que además dé soporte a la colaboración entre diversos usuarios para posibles futuros escenarios. Este editor debe basarse en el lenguaje visual de reglas ECA estudiado, de modo que la herramienta completa permite la especificación de reglas conforme a distintos grados de expresividad, en función de los conocimientos y experiencia de cada usuario. Además, gracias a la generalidad de este modelo de reglas la herramienta podría adaptarse para su uso en otros entornos reactivos, no quedando ligada únicamente al área de la Inteligencia Ambiental.

1.3. Organización del trabajo

El presente trabajo sigue la siguiente estructura: la sección 2 realiza un análisis de diversos trabajos relacionados con la personalización de comportamiento en ambientes inteligentes, estudiando sus puntos fuertes y las posibles mejoras a realizar. La sección 3 presenta un modelo de reglas ECA genérico y expresivo, que permitirá posteriormente la construcción de una herramienta basada en superficies interactivas para la especificación de comportamiento a través de dichas reglas, y que se detalla en la sección 4. La sección 5 presenta el experimento llevado a cabo para evaluar la usabilidad y comprensibilidad de la herramienta desarrollada comparando el rendimiento de dos grupos de usuarios, programadores y no programadores, frente a una serie de tareas propuestas, y se detallan los resultados obtenidos. Finalmente, en la sección 6 se realiza un resumen de las principales conclusiones obtenidas tras el experimento, y se resaltan diversas líneas de trabajo futuro.

2. Personalización de comportamiento en ambientes inteligentes

El desarrollo de mecanismos para la especificación de comportamiento por parte de usuarios no expertos para personalizar entornos inteligentes de acuerdo a sus preferencias ha sido objeto de una intensa actividad de investigación que pasamos a describir a continuación.

En *(Weis et al., 2006)* se presenta una herramienta de programación visual denominada Nexel, que permite personalizar aplicaciones mediante la configuración de los componentes involucrados. Esta herramienta se enmarca dentro de un proceso de desarrollo basado en componentes para aplicaciones pervasivas, que utiliza como mecanismo para la definición y despliegue de componentes el sistema PCOM *(Becker et al., 2004)*. Según este proceso de desarrollo, en primer lugar se debe crear los componentes necesarios para el entorno, y posteriormente en la etapa de configuración se construyen comportamientos que liguen las características de diversos componentes para crear otros más complejos y desplegarlos fácilmente. El editor visual que provee la herramienta, mostrado en la Figura 1 y Figura 2, ofrece al usuario la posibilidad de especificar qué acciones deben ser ejecutadas como respuesta a la ocurrencia de un determinado evento, o bien a raíz de la señal que ha lanzado un dispositivo determinado. Las acciones o procesos a ejecutar se definen arrastrando y conectando elementos gráficos dentro del área de especificación de la herramienta. Cada elemento gráfico tiene una semántica asociada, y existen construcciones de programación tales como iteraciones, bifurcaciones y secuencias de control expresadas mediante estos elementos gráficos. A pesar el potencial de este trabajo, está orientado a desarrolladores y no se considera adecuado para usuarios finales.

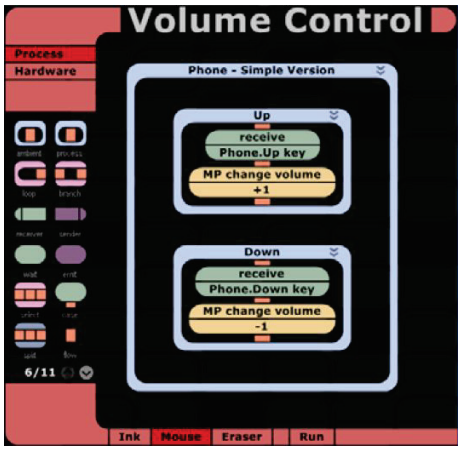


Figura 1. Interfaz de Nexel creando un componente para el control de volumen.

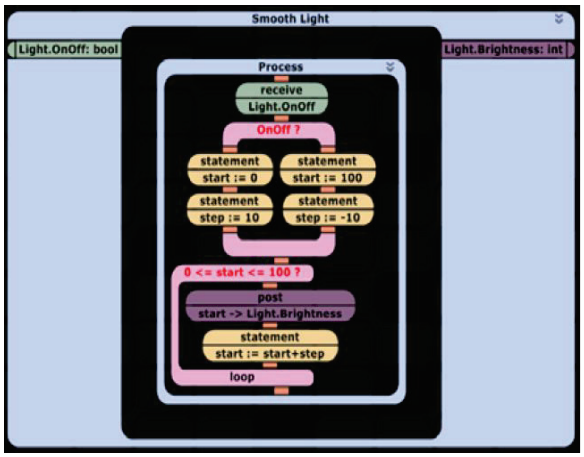


Figura 2. Interfaz de Nexel para el proceso de control de la luz de una lámpara.

Por su parte, aCAPella *(Dey et al., 2004)* es una aplicación sensible al contexto basada en la programación por demostración, que combina tanto la entrada directa del usuario como técnicas de *machine learning*. El sistema permite grabar entradas de distintos sensores, de modo que la información captada por los sensores se muestra más tarde en la interfaz (ver Figura 3) para que los usuarios filtren aquellas entradas correspondientes a una situación o condición determinada. Una vez filtradas las condiciones deseadas, se pueden establecer las acciones a ser ejecutadas. De este modo, los usuarios finales pueden probar el sistema, así como refinar los comportamientos especificados entrenando al sistema con nuevas repeticiones de las mismas condiciones, a fin de obtener mejores respuestas. Sin embargo, a pesar de ser un sistema comprensible por usuarios sin conocimientos de programación, la interfaz no resulta muy natural. Otro problema es la necesidad de que la situación deba producirse repetidas veces para poder automatizar la respuesta del sistema ante la misma obteniendo alta fiabilidad, ya que esto en ocasiones no es adecuado ni viable, por ejemplo para casos que rara vez deban tenerse en cuenta, como comportamiento del sistema ante robos, etc. Un sistema basado en reglas permitiría definir comportamiento para situaciones de rara ocurrencia sin necesidad de que éstas deban producirse.

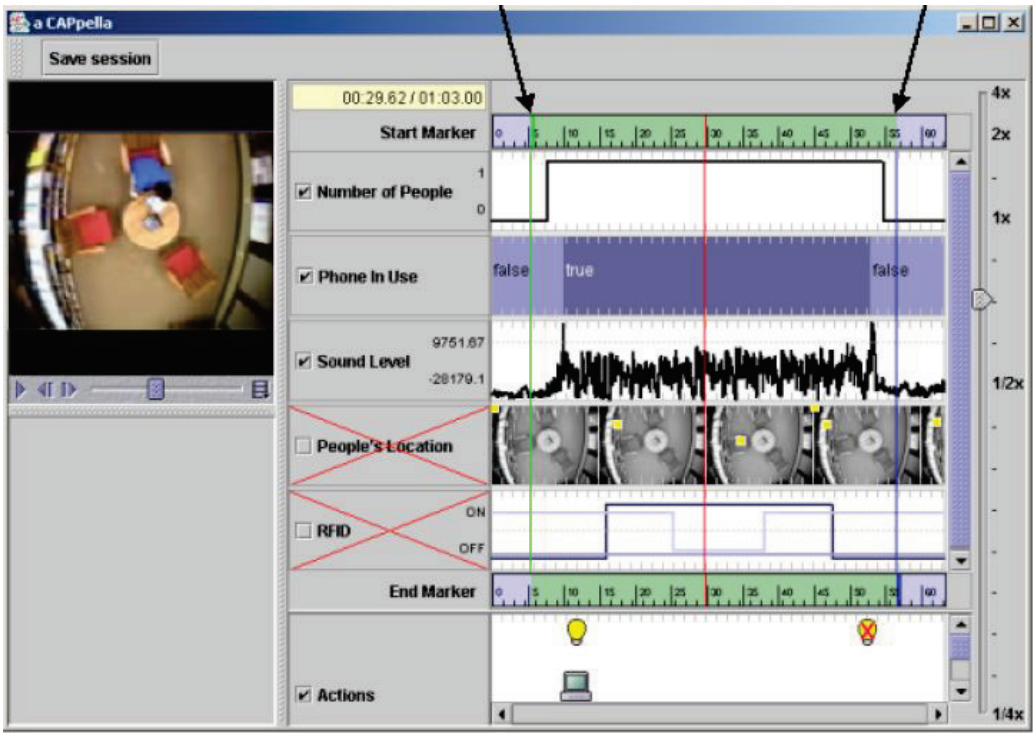


Figura 3. Interfaz de usuario del sistema aCAPella.

Otros trabajos han preferido emplear sistemas basados en reglas, como es el caso de *(Zhang & Brügge, 2004)*, donde se presentan dos herramientas de edición gráfica para la especificación de comportamiento en sistemas sensibles al contexto. La expresividad permitida para las reglas de esta propuesta se basa en construcciones IF-THEN básicas, donde tanto condiciones como acciones pueden combinarse únicamente mediante la conjunción (*and*). Cada condición implica la comparación entre dos elementos. En la parte de las acciones a ejecutar, las

acciones permitidas son la invocación de servicios sencillos de los dispositivos disponibles. El primero de los dos editores desarrollados (ver Figura 4) se basa en una interfaz de botones y ventanas tradicional, y hace uso de toda la expresividad del lenguaje de reglas ECA propuesto. El otro editor está destinado a dispositivos PDA y utiliza la metáfora de las piezas de un puzle e interacciones *drag-and-drop* para modificar reglas existentes (ver Figura 5). Los estudios realizados indican que la primera interfaz resulta demasiado compleja, mientras que la ayuda visual que se provee en la interfaz para PDAs facilita a los usuarios una mayor comprensión de las reglas que se editan, reduciendo además el número de errores cometidos. Sin embargo, estas dos herramientas de edición se han probado únicamente con usuarios programadores, pero se espera que sean adecuadas para usuarios no programadores.

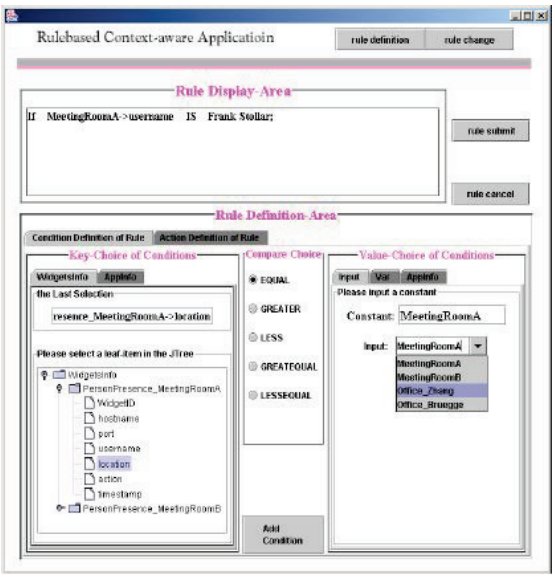


Figura 4. Interfaz para la construcción de reglas ECA basada en controles tradicionales.

Figura 5. Interfaz para la construcción de reglas ECA en PDAs empleando la metáfora del puzle.

Dey y Sohn presentan iCAP *(Dey et al., 2006)*, una herramienta para el prototipado rápido de sistemas sensibles al contexto que no requiere de la escritura de código. En iCAP el comportamiento se especifica en forma de reglas IF-THEN. La parte IF de la regla permite la conjunción (*and*) y disyunción (*or*) de condiciones, y cada una de las condiciones puede utilizar operadores de comparación como "*mayor que*", "*menor que*", "*igual*", etc. Únicamente se permite la conjunción de acciones. La herramienta está diseñada para ser sencilla de utilizar por usuarios finales del sistema, a la vez que pretende alcanzar la expresividad necesaria para llevar a cabo prototipos sencillos de aplicaciones sensibles al contexto, ya que es posible simular las reglas editadas antes de su despliegue. La interfaz está dividida en tres partes, visibles en la Figura 6: una para la exploración de elementos y definición de valores, otra para editar la parte condicional de la regla, y otra para la edición de las acciones. Se emplean menús circulares e interacciones mediante *drag-and-drop* ya que la herramienta está pensada para ser manipulada mediante un lápiz táctil. El estudio conducido para evaluar la

usabilidad del sistema resultó satisfactorio, ya que usuarios no programadores fueron capaces de especificar los escenarios propuestos sin problemas. iCAP es una herramienta con gran potencial, ya que permite especificar reglas IF-THEN sencillas y reglas basadas en relaciones espaciales. Sin embargo, las primeras no permiten expresar operaciones aritméticas, p. ej., poner el volumen de la radio a la mitad del volumen de la tele, mientras que la interfaz gráfica para expresar relaciones temporales no resulta muy intuitiva.

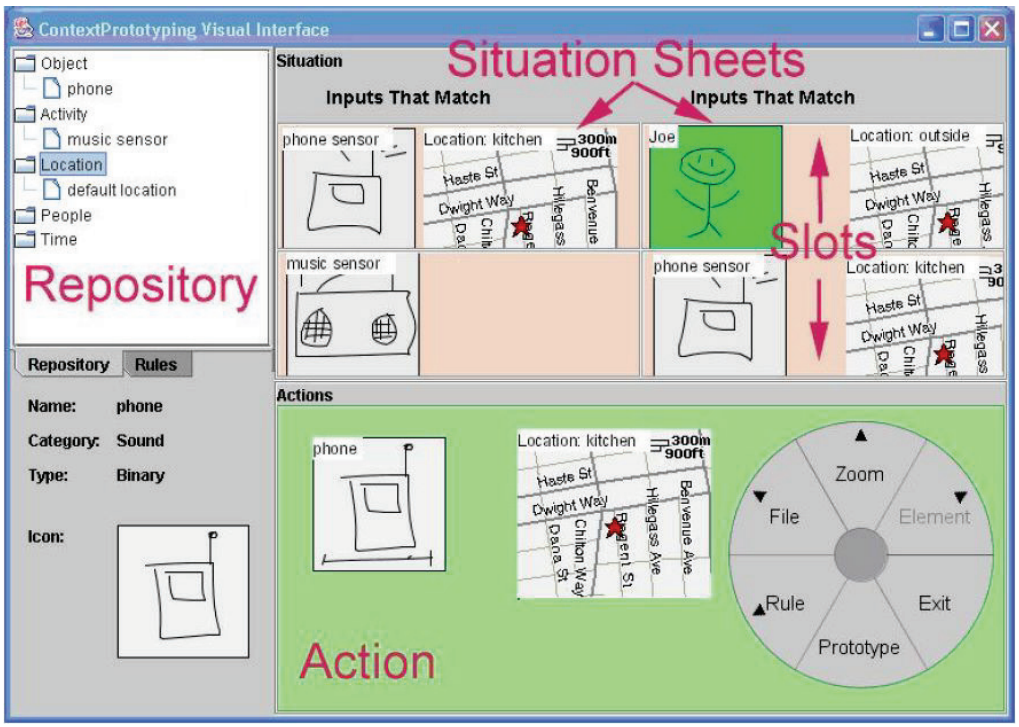


Figura 6. Interfaz de usuario de iCAP para la definición de reglas reactivas.

El trabajo de *(Beckmann & Dey, 2003)*, SiteView, utiliza un sistema de creación de reglas tangible, que si bien no alcanza la expresividad necesaria para ser utilizado en un entorno real y dinámico, sienta las bases del uso de la tecnología de las TUIs para la problemática de la definición de comportamiento en entornos inteligentes. El entorno de trabajo de SiteView puede verse en la Figura 7. Se dispone de un conjunto de tangibles que permiten representar las condiciones y acciones que el sistema es capaz de gestionar, una representación en miniatura del entorno inteligente considerado donde se situarán los tangibles asociados a las acciones, así como tres dispositivos para la edición de condiciones. Una pantalla de ordenador muestra tanto la regla que se está creando como el conjunto de reglas aplicable bajo las condiciones especificadas con los tangibles, y otra pantalla muestra qué efectos tendría la regla creada en el entorno inteligente. Las reglas de SiteView son construcciones IF-THEN básicas, que se crean de la siguiente manera: para especificar las condiciones ante las que se activa la regla, se deben colocar los tangibles de dichas condiciones sobre los dispositivos de composición de condiciones. La acción a realizar se indicará situando el tangible correspondiente a dicha acción sobre la representación en miniatura del entorno. Si bien se trata de una interfaz intuitiva, tal y como demuestran los estudios realizados, existen

desventajas en cuanto al tipo de reglas que SiteView permite definir. En primer lugar, únicamente se pueden indicar un máximo de tres condiciones por regla, y dichas condiciones tan sólo pueden combinarse mediante la conjunción. Por otra parte, tanto las acciones como las condiciones disponibles están prefijadas y son incómodas de extender, ya que cada condición o acción implica un tangible nuevo, de modo que deben ser acciones o situaciones sencillas para poder expresarlas con tangibles, y cada nueva situación implicaría la construcción del tangible asociado.

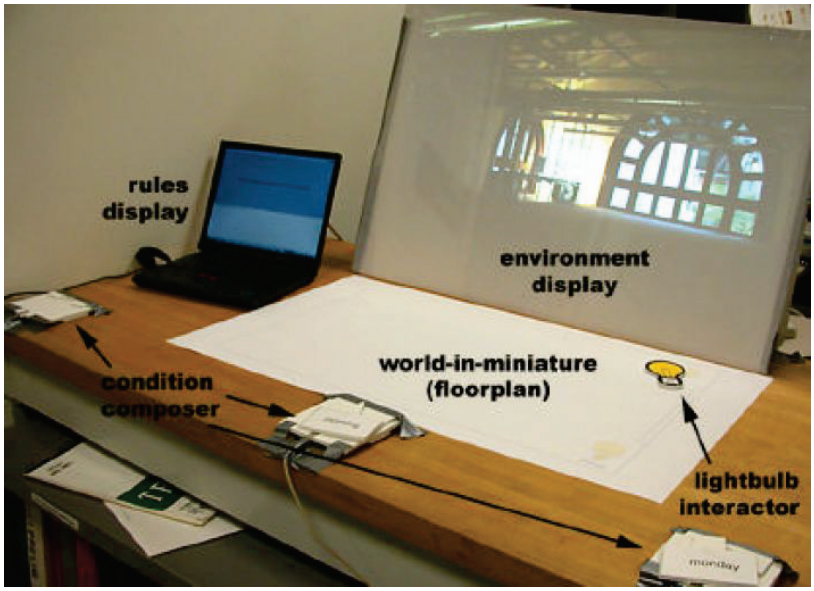


Figura 7. Entorno de creación de reglas provisto por SiteView.

Se pueden encontrar propuestas como *(Bonino et al., 2011)*, donde es posible especificar reglas más expresivas que las de los trabajos mencionados anteriormente. En este caso, la edición de reglas se llevaría a cabo por medio de una interfaz web orientada a usuarios no expertos, con reglas que sigan una estructura IF-WHEN-THEN. Esta estructura permitiría especificar diversos eventos en la parte IF, en forma de disyunción, de modo que la regla en cuestión se activaría ante la ocurrencia de cualquiera de ellos. Para cada evento, se puede indicar opcionalmente con una construcción WHEN la condición o condiciones bajo la que debe suceder dicho evento. Finalmente, se especificarían un conjunto de acciones a llevar a cabo. Hasta la fecha, este sistema continúa siendo una propuesta para la que todavía no se ha desarrollado un prototipo funcional, ni se han realizado evaluaciones con usuarios reales. Además, la gramática no ha sido definida formalmente.

En *(García-Herranz, Haya, et al., 2010)* se propone un sistema de programación independiente de la aplicación, basado en reglas ECA con alta expresividad. Este sistema permitiría a los usuarios finales especificar comportamiento complejo en entornos ubicuos. El punto central del sistema es un lenguaje textual para la especificación de reglas genérico y altamente expresivo. Una de las características que dotan de tal expresividad al lenguaje es la existencia de caracteres comodín que permitirían filtrar objetos de un conjunto. Otra de las incorporaciones de este lenguaje que no era posible especificar en los sistemas anteriores es la posibilidad

de establecer cuándo se ejecutan las reglas, es decir, si se trata de reglas que deben activarse periódicamente, o bien antes o después de un determinado instante de tiempo, o bien durante un intervalo de tiempo específico. En base a esta gramática para la definición de reglas se han desarrollado diferentes prototipos de interfaces de usuario que permiten adaptar el sistema a distintos tipos de usuarios y contextos de uso, abstrayendo en mayor o menor medida los aspectos complejos del lenguaje en función de los conocimientos del usuario final del sistema. La Figura 8 muestra una de las interfaces desarrolladas, basada en ventanas y controles tradicionales, que no está orientada a usuarios finales ya que hace visible la expresividad del sistema sin simplificar los conceptos. Una de las interfaces desarrolladas que resulta más adecuada para usuarios no programadores se basa en la metáfora de conectores magnéticos. Para ello se dispone de diversas zonas en la interfaz, cada una de ellas representando una parte de la regla, y los elementos disponibles se deben arrastrar al área correspondiente para crear construcciones sintácticamente correctas. Además, a medida que se añaden elementos al área de edición, aparecen o desaparecen elementos del área de exploración en función de las acciones disponibles, a fin de evitar errores semánticos. Sin embargo, esta interfaz más intuitiva únicamente se ha probado con una ontología limitada y restringe demasiado la funcionalidad del sistema, por tanto deben realizarse más estudios para obtener una interfaz adecuada que resulte comprensible y que no limite las capacidades del sistema.

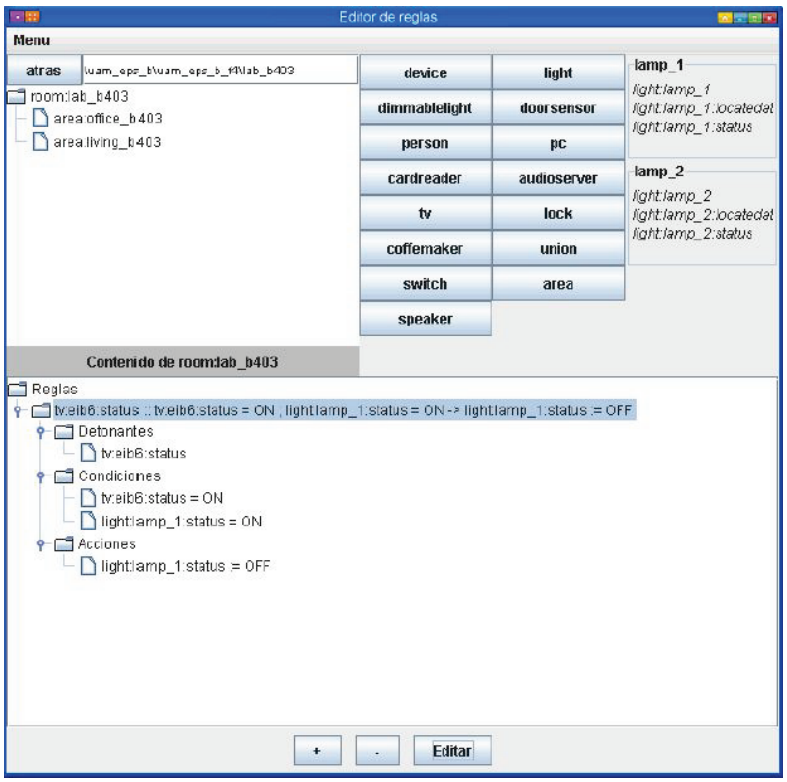


Figura 8. Una de las interfaces propuestas por García-Herranz para la creación de reglas ECA altamente expresivas.

Por otra parte, el trabajo presentado en *(Maternaghan & Turner, 2011)* consiste en un sistema orientado a mejorar la tele-asistencia de personas con necesidades

especiales, tales como personas de avanzada edad, para permitir que permanezcan en sus propias casas recibiendo la atención médica necesaria. Se trata de un campo de la inteligencia ambiental que requiere de una alta personalización, ya que cada usuario tendrá unos problemas y necesidades muy concretos. Por tanto, el sistema que proponen los autores, denominado *Homer*, debe ser extensible y adaptable. Para ello, se ha desarrollado una arquitectura basada en componentes y servicios, y un *framework* de desarrollo. Los desarrolladores de dispositivos deberán utilizar el *framework* provisto por *Homer*, basado en OSGi, para especificar qué condiciones, eventos y acciones pueden soportar sus dispositivos, entendidos como componentes en este trabajo. Estos dispositivos y servicios pueden invocarse por medio de *políticas*, que no son más que reglas de comportamiento reactivas que siguen una estructura WHEN-DO (equivalente a IF-THEN). Puesto que el lenguaje natural introduce ambigüedad, tanto condiciones como eventos se incluyen en la parte WHEN de la regla, mientras que las acciones a ejecutar se sitúan en la parte DO. Las condiciones y/o eventos de la regla pueden combinarse empleando los operadores *and*, *or* y *then*, este último permitiendo especificar una secuenciación de las condiciones, p. ej., "*when the door opens and then the light is on*". Las acciones únicamente pueden combinarse empleando expresiones *and*, pero se pueden especificar condiciones sobre las acciones definidas. Se han desarrollado dos interfaces de usuario, una para móvil y otra para tabletas interactivas (ver Figura 9 y Figura 10), pero no se han llevado a cabo experimentos con usuarios finales para probar la usabilidad del sistema. Además, las acciones a ejecutar no permiten utilizar el estado de algunos dispositivos para modificar en consecuencia el de otros, únicamente se permiten expresiones triviales.

Figura 9. Interfaz gráfica de Homer para dispositivos móviles.

Figura 10. Interfaz gráfica de Homer para tabletas táctiles.

3. Metamodelo de reglas de comportamiento

Con el objetivo de obtener una aproximación comprensible para definir comportamiento permitiendo distintos grados de expresividad en función de los conocimientos y experiencia del usuario, este trabajo se basa en un modelo de reglas enriquecidas mediante flujos de datos que permiten la definición de comportamiento en entornos reactivos, tales como los espacios inteligentes. En esta sección se presenta dicho modelo de reglas, y se resume parte del trabajo desarrollado en *(Pons, 2012)(Pons et al., 2011)*. El metamodelo de reglas completo abarca algunos aspectos que no van a ser detallados en este trabajo. Una explicación detallada de los mismos puede encontrarse en *(Pons, 2012)*.

Si bien en el campo de AmI existen numerosos *middlewares (Becker et al., 2004)(Fuentes et al., 2006)* capaces de manejar la gran variedad de dispositivos presentes en los entornos inteligentes de hoy en día, definiendo ontologías propietarias, este trabajo se basa en una abstracción más flexible y genérica. Esta abstracción parte de la definición de categorías/tipos y elementos/entidades, de modo que el modelo permite integrar cualquier ontología específica que pueda subsumirse en esta generalización sin afectar a la manera en la que las reglas de comportamiento serán editadas para cada una de las ontologías. Además, permite que el modelo de reglas pueda ser utilizado en otros ámbitos que requieran de la definición de comportamiento reactivo, como por ejemplo el área de juegos virtuales para fomentar la creatividad *(Catalá, García, Pons, et al., 2012)*.

3.1. Modelado del entorno: ecosistemas reactivos

Los entornos reactivos en los que los elementos del entorno responden a entradas de los usuarios o a cambios producidos en el sistema son considerados de manera genérica como ecosistemas en nuestra aproximación. Así pues, un espacio inteligente constituiría un ecosistema según el modelo propuesto. Estos ecosistemas están poblados con entidades, que no son más que una abstracción de los elementos inteligentes presentes físicamente en el espacio inteligente. Cada elemento de un espacio inteligente pertenece a una determinada categoría, de modo que en el ecosistema, cada entidad conformará a un determinado tipo de entidad, el cual permitirá definir las capacidades y características de las entidades pertenecientes a dicho tipo. A modo de ejemplo, pensando en una casa inteligente puede considerarse que algunos tipos de entidades serían *Televisión*, *Radio* o *AireAcondicionado*. Cada tipo de entidad del ecosistema define una serie de propiedades y acciones, p. ej., el tipo *AireAcondicionado* tendrá distintas acciones disponibles, como *Encender*, *Apagar, EstablecerVelocidadVentilador*, *EstablecerTemperaturaActual*, etc., y propiedades como *TemperaturaActual* y *VelocidadVentilador*, entre otras. Cada una de las entidades de este tipo, p.ej., la entidad *ACSalón*, podrá ejecutar las acciones definidas por su tipo. Sin embargo, el

estado de cada entidad, entendido como el valor de sus propiedades, será diferente al del resto de entidades y podrá cambiar durante la evolución del ecosistema.

Así como los espacios inteligentes responden a cambios en el entorno, los ecosistemas evolucionan debido a la ocurrencia de sucesos o eventos. Estas ocurrencias de evento conforman a un tipo de evento determinado, y son lanzadas por las entidades durante su ciclo de vida, p. ej., cada entidad del tipo *SensorTemperatura* podrá lanzar eventos del tipo *TemperaturaCambiada* para anunciar periódicamente cambios en la temperatura de la habitación.

La Figura 11 muestra el modelo de clases genérico para definir ecosistemas reactivos, donde pueden encontrarse los elementos anteriormente comentados. Cabe destacar que tanto propiedades, parámetros y atributos serán elementos que en tiempo de simulación tomarán valores de un determinado tipo. A su vez, los tipos de entidades también forman parte de la colección de tipos disponible en el ecosistema. Por tanto, una propiedad podría tomar tanto valores numéricos como representar una entidad del ecosistema. Sin embargo, estas nociones resultan más técnicas, ya que el metamodelado de tipos se detalló en *(Pons, 2012)* y no es necesaria su explicación para comprender el trabajo que se comentará en las siguientes secciones de esta memoria.

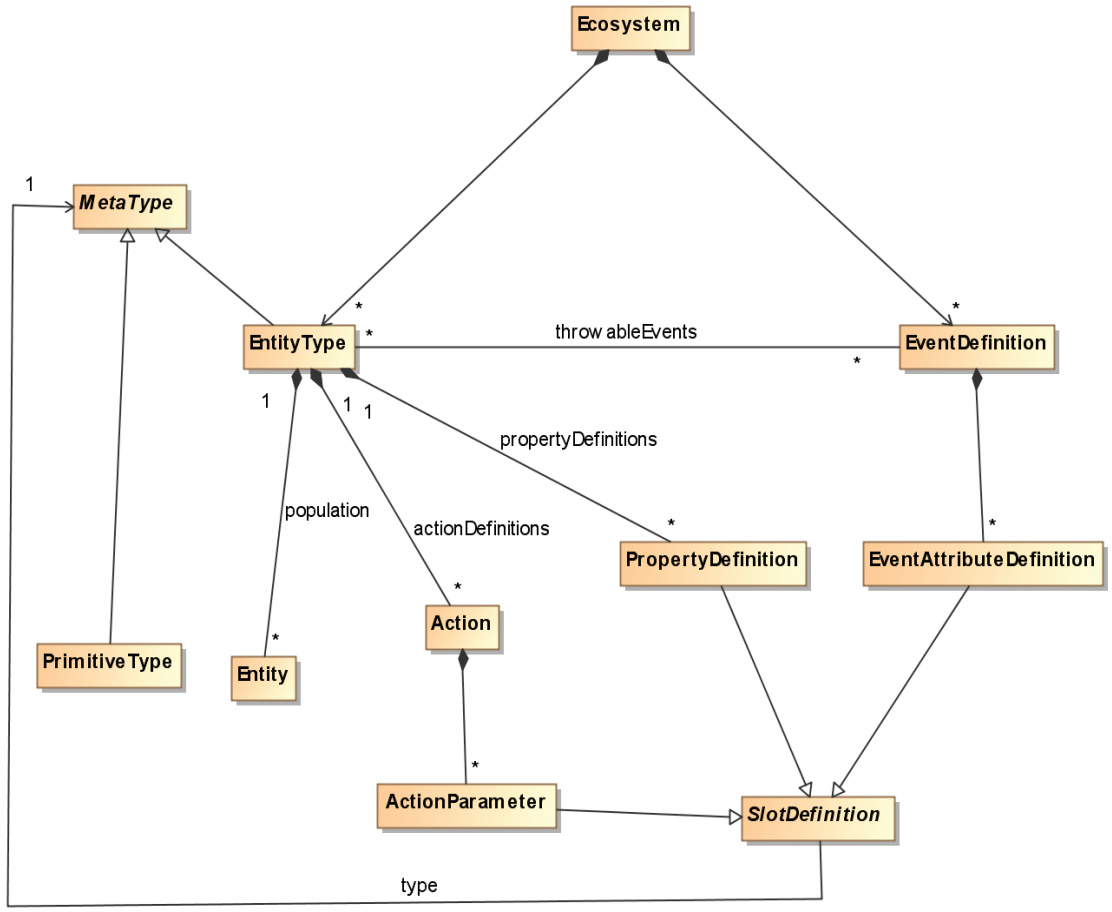


Figura 11. Diagrama de clases para la definición de ecosistemas reactivos.

3.2. Modelado del comportamiento: reglas ECA

Con estas abstracciones como pilares básicos del modelo, una regla se define formalmente como un par ordenado R = <P, Q>, donde P es el antecedente y Q es el consecuente. El antecedente P, definido como P = (E, S, C), está formado por la ocurrencia de un evento de un determinado tipo E, que ha sido provocado por una entidad de la población origen S, y además, tanto E como S deben cumplir una determinada condición C. El consecuente Q, definido como Q = (T, O, F, {DP}), está formado por una población de entidades destino T, una condición de filtrado F que permite filtrar las entidades de T a las que afectará la operación O, y un conjunto de procesos de datos {DP} que indican cómo se establecen los parámetros de O antes de su ejecución.

En busca de soluciones expresivas, las poblaciones origen y destino de la regla pueden especificarse de dos maneras diferentes. En primer lugar, puede tratarse de entidades específicas del ecosistema (p.ej. *ACSalón*) indicando un comportamiento individual que únicamente se aplica a la entidad en cuestión. En segundo lugar, las poblaciones pueden especificarse como colecciones de entidades del mismo tipo, ya que es común que distintas entidades de un mismo tipo se comporten igual en algunos escenarios (p.ej. todos los *AiresAcondicionados* de una misma habitación). Las condiciones C y F son muy útiles a la hora de refinar estas poblaciones colectivas si es necesario, evitando tener que definir una regla individual para cada entidad que atienda al mismo comportamiento.

La semántica para una regla R es la siguiente: cuando una entidad perteneciente a S lanza una ocurrencia de un evento E, y la condición C se satisface, entonces la regla es instanciada y activada. La operación O se ejecutará sobre aquellas

(a)

CUANDO S: *TeléfonoCasa* **NOTIFICA** el evento **E**: *RecibiendoLlamada*
[Y SE SATISFACE la condición **C**: *RecibiendoLlamada.llamador = "Mary"* **]**
ENTONCES PARA CADA entidad en **T**: *TVSalón*
[QUE CUMPLA la condición de filtrado **F**: *TVSalón.Encendida = Cierto* **]**
EJECUTAR O: *TVSalón.Volumen * 20 / 100 → TVSalón.Volumen*

(b)

CUANDO S: *Lugar* **NOTIFICA** el evento **E**: *PersonaEntra*
[Y SE SATISFACE la condición **C**: *Lugar.NumPersonas > 3* **]**
ENTONCES PARA CADA entidad en **T**: *AireAcondicionado*
[QUE CUMPLA la condición de filtrado **F**:
AireAcondicionado.Ubicación = PersonaEntra.Hacia **]**
EJECUTAR O: *AireAcondicionado.TemperaturaReferencia + Lugar.NumPersonas * 2 → AireAcondicionado.TemperaturaReferencia*

Figura 12. Ejemplos de reglas de comportamiento basadas en el modelo de reglas ECA propuesto.

entidades en T que pasen el filtro establecido con F. los parámetros de la operación a ejecutar se establecerán según la especificación de los procesos de datos {DP}. La Figura 12 muestra dos reglas ilustrativas del tipo de comportamiento que es posible definir con el modelo propuesto. Según la regla (a), cuando el teléfono de la casa recibe una llamada, y el llamador es Mary, entonces se establece el volumen de la televisión del salón al 20% de su volumen actual si dicha tele está encendida. Por su parte, la regla (b) especifica que cuando un lugar notifica la entrada de una persona en dicha estancia, y si el número de personas en la estancia a la que se entra es mayor que 3, entonces para cada aire acondicionado de dicha estancia, se debe establecer la temperatura de referencia según la expresión indicada. Nótese que en la regla (a) se ha hecho uso de poblaciones específicas, mientras que la regla (b) empleaba poblaciones como conjuntos de entidades para posteriormente filtrarlas.

Tanto las condiciones C y F, como la operación de la regla, se definen como procesos de datos en esta aproximación. Un proceso de datos no es más que la representación visual de una expresión de asignación. Las condiciones C y F son expresiones booleanas, cuyo resultado determinará si la condición en cuestión se satisface o no. El tipo de operaciones que se permite realizar en las reglas de este modelo son, o bien la modificación de una propiedad de la población destino, o bien la ejecución de una de las acciones disponibles de esta población. En el primer caso, es necesario un único proceso de datos para definir la operación, consistente en la expresión para el cálculo del nuevo valor de la propiedad a modificar. En el segundo caso, serán necesarios tantos procesos de datos como parámetros tenga la acción a ser ejecutada. Cada uno de estos procesos de datos corresponderá a la expresión con la cual se calculará dicho parámetro de la acción. En la sección 3.3 se detalla el lenguaje visual empleado para la definición de procesos de datos.

La Figura 13 muestra el diagrama de clases que permite definir reglas reactivas como las explicadas en este apartado. Una regla (*ReactiveRule*) consta de un tipo de evento (*EventDefinition*), una población origen (*SourcePopulation*) y una población destino (*TargetPopulation*), que podrán ser o bien entidades específicas (*EntitySourcePopulation* y *EntityTargetPopulation*) o bien conjuntos de entidades de un tipo (*EntityTypeSourcePopulation* y *EntityTypeTargetPopulation*). Además, la regla permite definir tres tipos de procesos de datos (*DataProcess*): uno para la condición (*PreconditionDataProcess*), otro para el filtrado (*FilterPreconditionDataProcess*), y los correspondientes a la operación. En función del tipo de operación (*OperationBinding*), que puede ser o bien la modificación de una propiedad (*PropertyBinding*) o la ejecución de una acción (*ActionBinding*), habrá uno o varios *BindingDataProcess*.

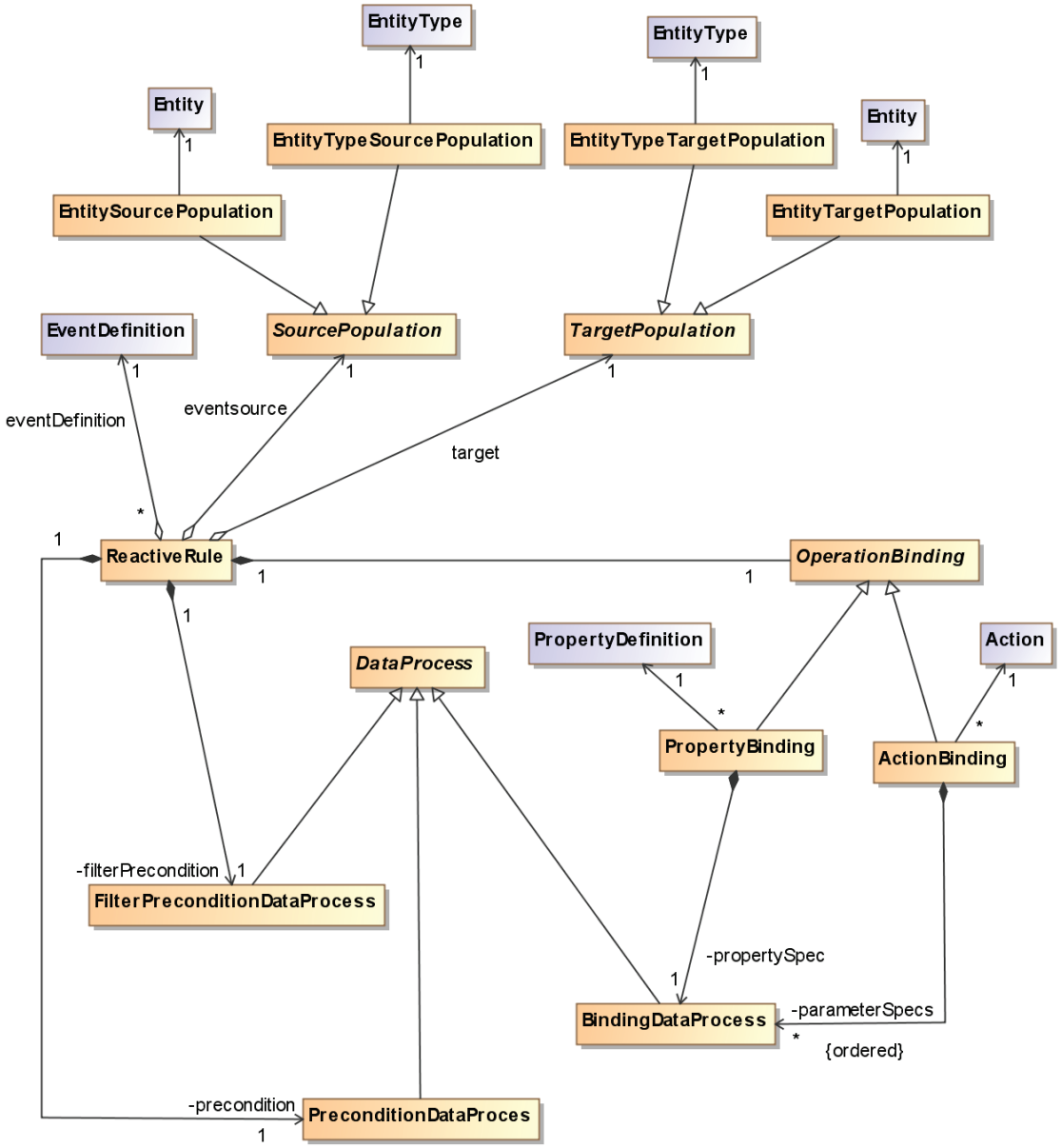


Figura 13. Diagrama de clases para dar soporte a la definición de reglas ECA.

3.3. Expresividad mediante flujos de datos

Tal y como se ha explicado anteriormente, los procesos de datos constituyen representaciones visuales de expresiones de diverso tipo, y por tanto, deben contener una serie de elementos gráficos que permitan definir los conceptos necesarios de la expresión. En primer lugar, un proceso de datos contiene un conjunto de elementos proveedores de datos, tales como propiedades de las entidades, atributos del evento, o constantes numéricas. En segundo lugar, es necesario disponer de operadores para transformar estos datos de entrada en datos nuevos. En tercer lugar, debe existir un elemento al que asignar el resultado de la expresión, que en función del tipo de la expresión esperará o bien un valor

booleano si se trata de una condición, o un valor numérico si se trata de una operación. Finalmente, es necesario establecer flujos de datos entre todos estos elementos para poder construir expresiones semánticamente correctas. Las conexiones mediante flujos de datos permiten la aplicación de operadores a los datos de entrada, así como la combinación entre operadores para crear expresiones complejas. El resultado final de la expresión del proceso de datos debe ser asignado a una propiedad, parámetro de acción o resultado de condición.

La Figura 14 presenta el diagrama de clases que permite la creación de procesos de datos como los presentados, donde existen distintos tipos de nodos, que disponen de puertos de entrada y/o salida a través de los cuales se realiza la conexión mediante flujos de datos. Puesto que las propiedades, atributos, parámetros y valores numéricos tienen un determinado tipo de datos, los puertos de entrada y salida de los operadores también deben definir el tipo de datos que aceptan y/o producen.

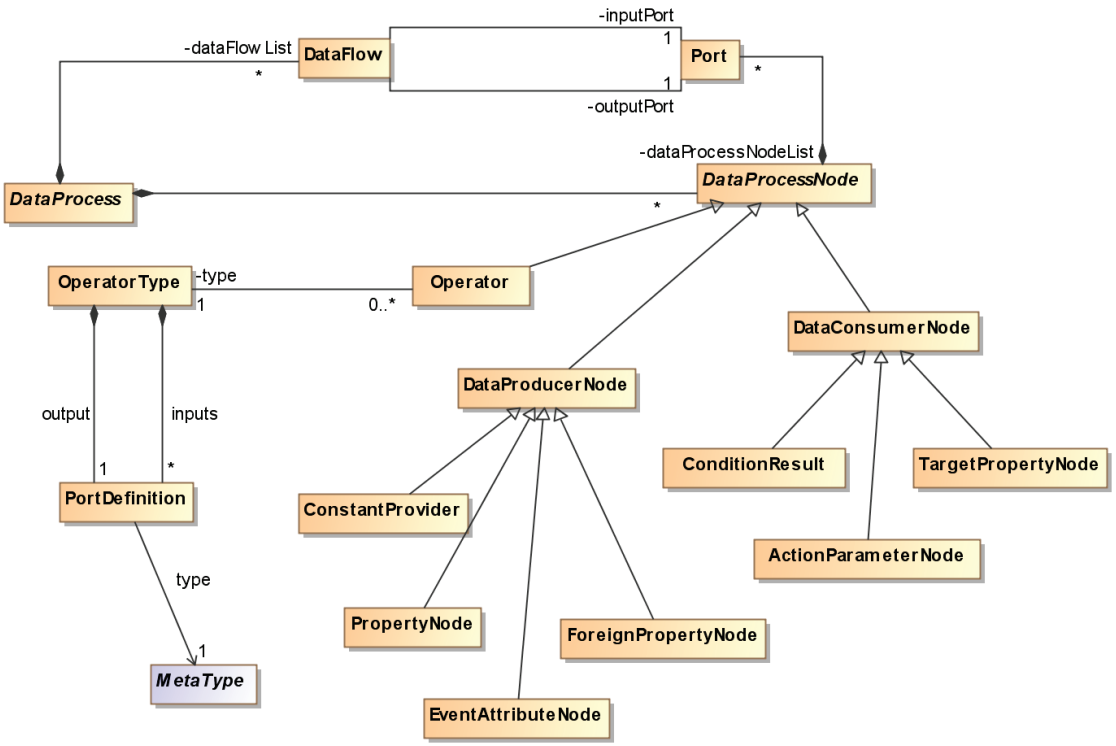
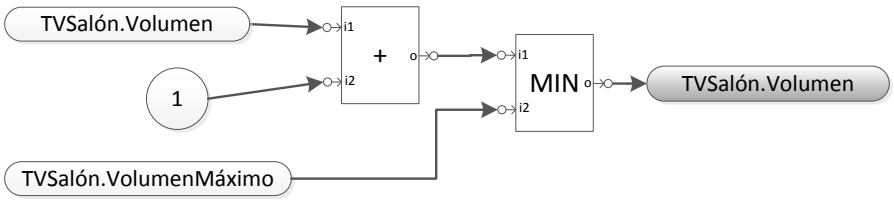


Figura 14. Diagrama de clases para la creación de expresiones visuales basadas en flujos de datos.

La Figura 15 muestra tres ejemplos de expresiones representadas de manera visual empleando conexiones de flujos de datos como las descritas. La (a) constituye una expresión básica para el incremento del volumen de un televisor hasta llegar al máximo, donde satura. La (b) es un proceso de datos para evaluar una condición, en concreto, si el nivel de la persiana de la cocina está en el rango definido. La (c) es un proceso de datos correspondiente a una operación, donde se modifica el valor de la propiedad *Intensidad* de la entidad *FocoCentralSalón*, según la expresión *MIN((Salón.NumPersonas * 10) + FocoCentralSalón.Intensidad, 100)*.

Para mayor información sobre la expresividad permitida con esta aproximación basada en flujos de datos, consultar *(Pons, 2012)*.

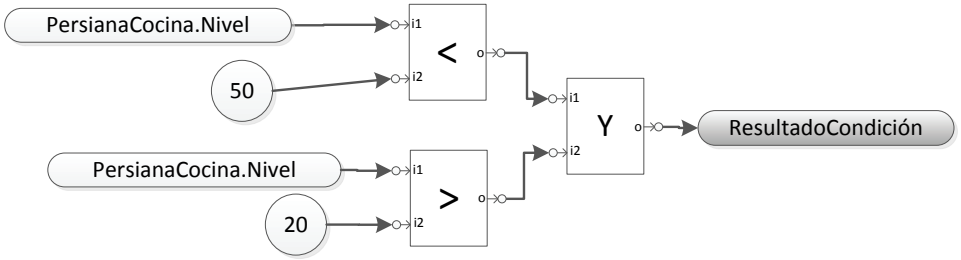
(a)



MIN (TVSalón.Volumen + 1, TVSalón.VolumenMáximo) → TVSalón.Volumen

---

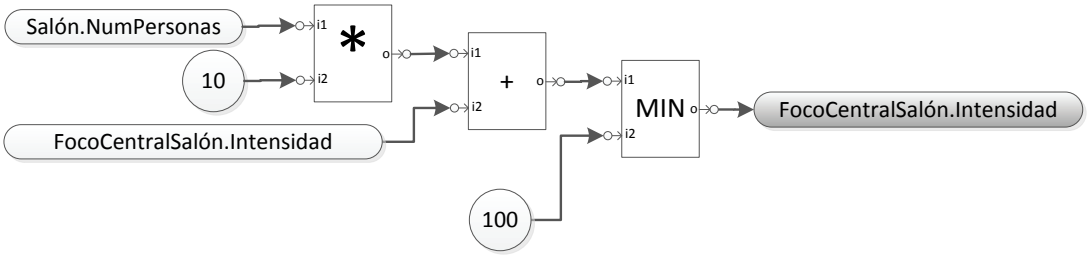
(b)



(PersianaCocina.Nivel < 50) Y (PersianaCocinal.Nivel > 20) → ResultadoCondición

---

(c)



MIN ((Salón.NumPersonas * 10) + FocoCentralSalón.Intensidad, 100) → FocoCentralSalón.Intensidad

Figura 15. Ejemplos de distintos procesos de datos expresados de manera visual, y su correspondiente expresión textual.

4. Edición de reglas

Se ha diseñado e implementado una herramienta para la edición de reglas de comportamiento basadas en el meta-modelo de reglas enriquecido mediante flujos de datos descrito en la sección 3. A partir de un prototipo de editor de reglas para una interfaz tradicional con botones y ventanas *(Pons, 2012)* se ha podido extraer algunos requisitos a tener en cuenta para esta primera versión del editor. La herramienta ha sido concebida para su uso en superficies interactivas, puesto que estas nuevas plataformas proveen mecanismos de interacción novedosos e intuitivos, tales como la interacción directa con los dedos o con elementos tangibles. Estas técnicas de interacción, además de resultar más cómodas y naturales para usuarios no expertos, son muy adecuadas para realizar la edición de flujos de datos, punto central del proceso de edición de comportamiento, y además posibilitan la futura exploración de la edición cooperativa de reglas de comportamiento.

4.1. Consideraciones previas

Un prototipo del editor se desarrolló en C# mediante una interfaz tradicional basada en controles y ventanas, empleando como plataforma un HP TouchSmart IQ522 dotado de pantalla táctil de 22' para hacer las pruebas de concepto. La interacción con el prototipo del editor se llevó a cabo de manera táctil, para obtener una experiencia lo más cercana posible a lo que se obtendría con la versión final para superficies interactivas presentada en este trabajo. La Figura 16

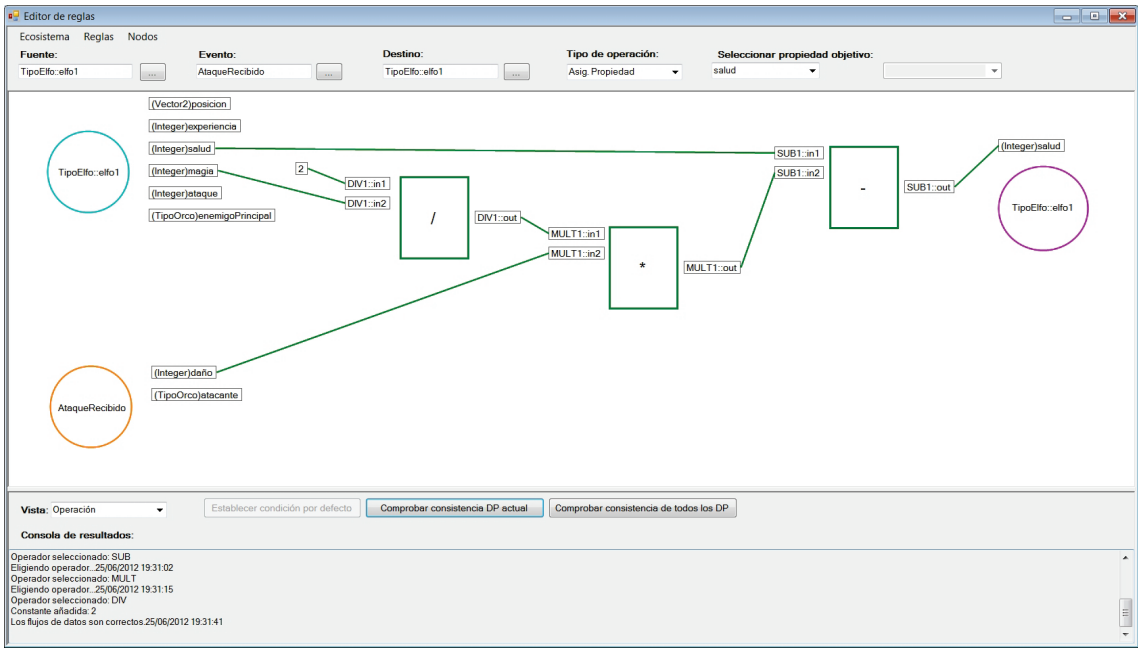


Figura 16. Prototipo de un editor de reglas de comportamiento para una interfaz basada en botones y ventanas.

muestra una captura de pantalla de la interfaz de esta herramienta, con una regla correctamente editada. Tras un uso exhaustivo del prototipo para la edición de reglas de comportamiento de diversa índole, ha sido posible detectar determinados requisitos de diseño que facilitarían la tarea de edición sobre superficies interactivas.

En primer lugar, ya en el prototipo del editor se optó por llevar a cabo la tarea de edición de comportamiento siguiendo un proceso iterativo e incremental. Por tanto, la edición de una regla de comportamiento se ha diseñado como una sucesión de decisiones a ser tomadas en diferentes vistas parciales de la regla que se muestran a los usuarios separadamente. Con ello se pretende, por un lado, paliar los problemas de limitación de espacio del área de edición, y por otro lado, permitir que los usuarios se centren únicamente en un único proceso de datos simultáneamente. Esta decisión de dividir el proceso de edición en vistas se ha mantenido en el editor que se presenta en este trabajo ya que aporta grandes ventajas. Así pues, en cada vista los usuarios deberán emplear tanto sus dedos como tangibles específicos para acceder a las colecciones deseadas e ir editando el proceso de datos de la vista en cuestión.

Por otra parte, se detectó que el espacio disponible para la edición debía ser lo más reconfigurable posible, permitiendo al usuario redistribuir los elementos a lo largo del área de edición según sus necesidades de espacio. Los elementos principales de la regla, a saber origen, evento y destinatario, permanecían en posiciones fijas en el prototipo, y eran visibles durante todo el proceso de edición. Para el editor que se presenta no se ha considerado necesario que dichos elementos sean visibles constantemente, ya que hay determinadas vistas en las que pueden no ser requeridos, y mantenerlos en el área de edición consume espacio que podría ser aprovechado para ubicar otros elementos más necesarios. Sin embargo, corresponde al usuario tomar la decisión de qué elementos desea ver en cada vista, y las colecciones asociadas a estos elementos deben estar disponibles en cualquier momento para su visualización.

Otra cuestión a tener en cuenta en cuanto a la limitación de espacio concierne a las propiedades y atributos del origen y el evento de la regla, respectivamente. En el prototipo del editor, tanto las propiedades del origen como los atributos del evento permanecían fijos en el área de edición, listos para ser utilizados en el proceso de datos en cualquier momento. Pero esto no hace más que consumir, de nuevo, espacio del área de edición, siendo que tan sólo unas pocas propiedades o atributos van a ser necesarios en cada proceso de datos. De este modo, para la versión definitiva del editor, propiedades y atributos se añaden al área de edición según se requieran. Esto conllevará aumentar el número de interacciones, pero hace más legible la regla y más cómodo el proceso de edición introduciendo menos ruido.

Ya en el prototipo se disponía de varios mecanismos de comprobación de errores, que se han seguido manteniendo en la versión tangible del mismo, y que ofrecen al usuario *feedback* visual acerca de los errores que ha cometido en el proceso de edición. Si bien en el prototipo se trataba de comprobaciones *off-line*, que debían realizarse a petición del usuario pulsando un botón, para el editor sobre una superficie interactiva se ha tratado de automatizar esta

retroalimentación visual todo lo posible de modo que los errores sean detectados lo antes posible por el usuario para evitar sucesivas iteraciones en la definición de la regla.

Puesto que las superficies interactivas permiten la edición colaborativa situando a varios usuarios alrededor de la misma mesa y compartiendo el espacio de trabajo, el editor que se presenta ofrece controles 360º y los textos aparecen reflejados en dos orientaciones para facilitar su lectura, de modo que la herramienta está preparada en caso de plantearse escenarios donde la edición colaborativa resulte interesante. Un posible escenario podría ser, por ejemplo, una sala inteligente de juegos para niños, donde varios niños se encuentran alrededor de la mesa interactiva para tratar de configurar a su gusto el color de las luces de la sala en función de la actividad que estén realizando.

4.2. Modelo de diseño

Previa implementación de la herramienta de edición de reglas para superficies interactivas, se llevó a cabo el modelado de la interfaz del sistema. De este modo se ha obtenido un sistema altamente mantenible, adaptable a futuros cambios fruto de modificaciones o rediseño de los controles. Además, este diseño permite reutilizar componentes gráficos para otros usos fuera de la herramienta de edición, como son las conexiones de nodos mediante flujos de datos.

La Figura 17 muestra el diagrama de clases empleado en la implementación de la interfaz gráfica del editor de reglas. Cualquier aplicación (*App*) a ejecutar tendrá un conjunto de controles (*Control*) encargados de recibir y notificar las interacciones de los usuarios. Estos controles pueden ser elementos gráficos (*WheelMenu*, *NumericalInsertionControl*, *HaloSpot*) o controles sin ninguna representación visual asociada, con lo que únicamente se encargarían de la gestión y manejo de interacciones y la invocación de servicios (*Canvas*, *AGraphCanvas*). Las clases *RuleEditorApp* y *Canvas* son las encargadas de dirigir el flujo de control de la aplicación de edición de reglas. En concreto, una instancia de *RuleEditorApp* debe disponer de cuatro instancias del control *Canvas*, que permitirán manejar cada una de las vistas en las que se descompone una regla reactiva. A excepción de la vista inicial, el resto de vistas deben permitir la creación de expresiones visuales en forma de árbol o grafo. Para ello, se ha especializado el control *Canvas* en un nuevo control *AGraphCanvas*, que permitirá controlar la definición visual de expresiones. Para ello, una instancia de *AGraphCanvas* estará compuesta por controles de dos tipos especiales, *Edge* y *GraphNode*, además del resto de controles de los que pueda disponer un canvas normal. La clase *GraphNode* representa un nodo del grafo que constituye la expresión, y tendrá distintos puertos *AGraphPort* a los que poder conectar instancias de *Edge*. De este modo, un flujo de datos tendrá como representación visual una instancia de la clase *Edge*, que no es más que una línea entre dos puertos de dos nodos distintos. Las clases *GraphNode* y *AGraphPort* deberán ser especializadas en función del tipo de nodo o puerto en cuestión, ya que se dispone de nodos proveedores de datos y nodos consumidores de datos, cuyos comportamientos serán en ocasiones distintos.

Existen otros controles necesarios para la edición de reglas con la herramienta tangible. La clase *WheelMenu* se encarga de gestionar el acceso a las colecciones, y los elementos de una colección vienen representados por la clase abstracta *WheelMenuItem*. Ambas clases deberán especializarse en función del tipo de colección y elementos que vayan a contener, y de la información que deba mantenerse de los mismos. La clase *HaloSpot* permite disponer de un control que reacciona ante el posicionamiento de un tangible determinado sobre la superficie visible de este control. Su representación visual es un área circular de un determinado color, que parpadea periódicamente. Se empleará en la vista inicial de la regla para iniciar el proceso de edición. La clase *NumericalInsertionControl* ofrece un control en forma de regla para introducir constantes numéricas.

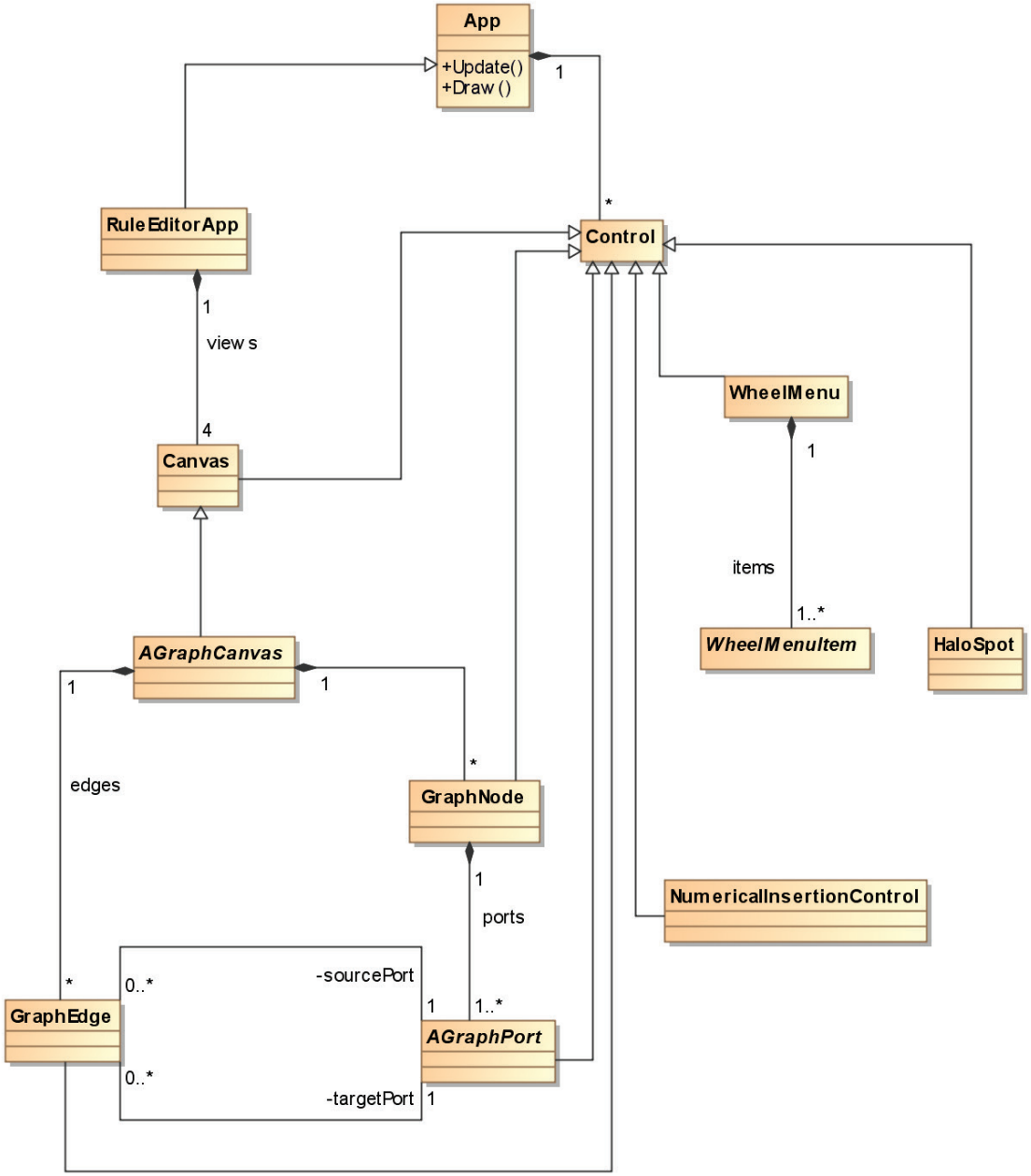


Figura 17. Diagrama de clases para la implementación del editor de reglas gráfico.

La integración de este modelo junto al modelo de reglas presentado en la sección 3 permite un diseño del sistema siguiendo un patrón Modelo-Vista-Controlador (MVC). La parte del Modelo corresponde al núcleo del sistema, es decir, el modelo de reglas ECA basado en flujos de datos. La parte de la Vista correspondería al conjunto de controles gráficos para la visualización de los elementos. Finalmente, la parte del Controlador la llevaría a cabo las clases *RuleEditorApp* y *Canvas*, clases sin visualización en la interfaz que únicamente se encargan de manejar los contactos recibidos y los servicios solicitados por los controles gráficos, traduciéndolas en la invocación de servicios en el modelo de reglas, de modo que los cambios en la interfaz gráfica tengan su correspondencia en el sistema de gestión de reglas (crear flujos de datos, añadir operadores, establecer el origen de una regla, etc.).

4.3. Implementación

El editor de reglas *(Pons, 2013)* que se propone en esta sección ha sido desarrollado para una unidad *Microsoft PixelSense 1.0*[1] (ver Figura 18). Esta mesa interactiva permite captar las interacciones del usuario con la superficie de la mesa gracias a un sistema de visión de 5 cámaras infrarrojas y un proyector, empleando principios de iluminación difusa *(Schöning et al., 2008)*. Los contactos que la mesa es capaz de reconocer son dedos o etiquetas, y en todo caso se provee información sobre la posición, rotación, etc., de los dos tipos de contacto.

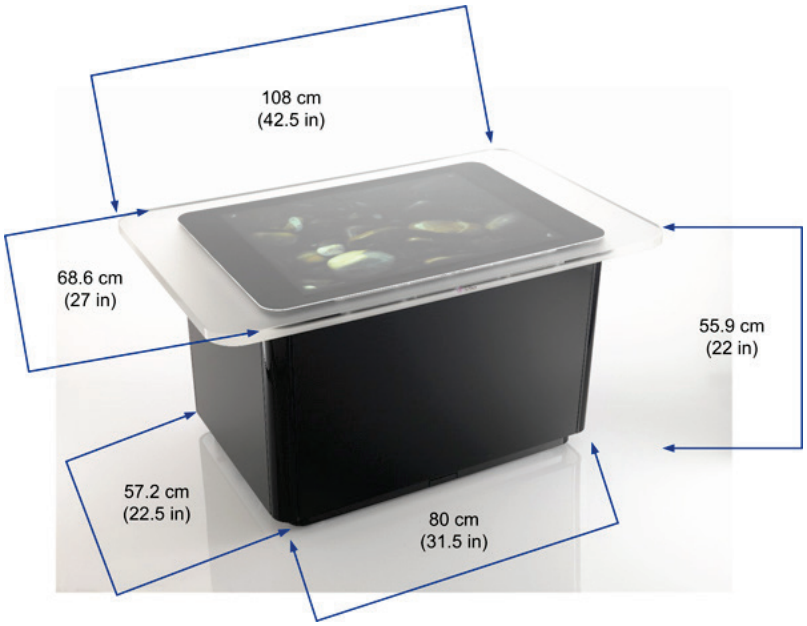


Figura 18. Dimensiones de la mesa interactiva Microsoft PixelSense 1.0[2].

[1] Anteriormente conocida como Microsoft Surface 1.0 (http://www.microsoft.com/en-us/pixelsense/gettingstarted10.aspx)
[2] Imagen extraída de http://technet.microsoft.com/en-us/library/ee692114%28v=surface.10%29.aspx

La pantalla es de 30 pulgadas, cuenta con una resolución de 1028x768 píxeles y el área de edición de la regla abarca la pantalla completa del dispositivo. Se han empleado el framework XNA de Microsoft y el Microsoft Surface SDK v1.0 para la implementación de los controles y la detección de interacciones. Además, las etiquetas se han integrado con elementos tangibles para permitir un mejor manejo de los mismos, tal y como muestra la Figura 19. Las siguientes subsecciones describen los componentes principales del editor y detallan cómo se lleva a cabo el proceso completo de edición.

Figura 19. Tangibles con etiquetas.

4.3.1. Exploración de colecciones

El editor de reglas de comportamiento que se presenta ha sido diseñado para que la interacción se realice completamente con los dedos, empleando controles diseñados específicamente para su uso en superficies interactivas. Puesto que la edición de una regla conlleva la exploración y selección de diferentes colecciones, se ha hecho uso de un control genérico para la manipulación de colecciones sobre superficies interactivas desarrollado por el grupo de investigación ISSI *(Catalá, García, Jaen, et al., 2012)*. Dicho control facilita la tarea de la exploración de las distintas colecciones necesarias para la definición de una regla, puesto que mediante estas colecciones el usuario será capaz de seleccionar y extraer propiedades, atributos, operadores y demás elementos que conformarán un proceso de datos.

Tal y como se explicó en la sección 3, hay tres elementos principales en una regla: la población origen, el evento que activa la regla, y la población destino que se ve afectada. Para seleccionar tales elementos, el editor provee tres colecciones, cada una de ellas asociada a uno de dichos elementos. Estas colecciones y los elementos que contienen son accedidos de manera estructurada, es decir, que se dispone de diferentes niveles dentro de la colección que deberán ir explorándose hasta llegar al elemento deseado. La estructuración en niveles permite organizar los elementos de la colección atendiendo generalmente a su tipo, de manera que

cada selección en un nivel permite filtrar los elementos que se mostrarán en el nivel siguiente.

Por ejemplo, suponiendo que un usuario debe establecer como población origen de la regla la entidad *ACSalón*, la primera decisión que debería tomar en la colección asociada sería si se trata de una categoría de elementos (tipo de entidad) o de un elemento concreto (entidad). Puesto que *ACSalón* es una entidad, se debería navegar por la colección de entidades concretas. Así pues, a continuación se mostrarían los tipos de las entidades del ecosistema como forma de agrupación de las mismas. Seleccionando el tipo de entidad *AireAcondicionado*, el siguiente nivel de la colección mostraría todas las entidades de este tipo (*ACOficina*, *ACComedor*, *ACDormitorio*, etc.), entre las cuales se encuentra la entidad *ACSalón*. Al seleccionar dicha entidad, esta quedaría establecida como la población origen de la regla, mostrándose sus propiedades para hacerlas disponibles en el momento en que se requieran durante la edición posterior de un proceso de datos de la regla. En cuanto a la selección de la población destino, el proceso a seguir sería el mismo. En lo concerniente al evento de la regla, la selección es mucho más simple. Tal y como se explicó en la sección 3, cada tipo de entidad determina el conjunto de tipos de evento que sus entidades son capaces de producir. De este modo, en función de la población origen seleccionada, el conjunto de eventos susceptibles de producirse varía. Por tanto, la colección de evento se actualiza dinámicamente para mostrar al usuario únicamente aquellos eventos que pueden ser lanzados por la población que se ha fijado como origen de la regla. Al seleccionar uno de estos tipos de evento, éste quedará establecido como disparador de la regla, y la colección mostrará los atributos que tenga asociados para que puedan emplearse posteriormente en la edición de un proceso de datos.

Existen dos colecciones más, la primera para seleccionar operadores y añadirlos a los procesos de datos, mientras que la segunda contiene elementos ajenos, entendidos como entidades o propiedades que no forman parte de las poblaciones origen y destino, pero que sin embargo resultan necesarios para la definición de la regla. En primer lugar, la colección de operadores contiene el conjunto de operadores disponibles para transformar los datos de un proceso de datos. Los operadores de esta colección se han agrupado en cuatro categorías: operadores lógicos, operadores relacionales, operadores aritméticos, y finalmente selectores. Por tanto, el primer nivel de la colección de operadores muestra los nombres de las categorías de operadores disponibles, de modo que el usuario debe seleccionar la categoría que desee para mostrar los operadores que contiene la misma, y finalmente poder seleccionar alguno para arrastrarlo al área de edición. La Figura 20 muestra a un usuario seleccionando el operador "*mayor que*" para arrastrarlo al área de edición del proceso de datos. Por su parte, la colección de elementos ajenos dispone de todas las entidades y tipos de entidades del ecosistema, tal y como sucedía en las colecciones del origen y del destinatario, de modo que el método de exploración de estas colecciones es similar. Sin embargo, la selección de un elemento en la colección de entidades ajenas no supone su establecimiento ni como origen ni como destinatario de la regla, sino que sencillamente permite utilizar cualquiera de las entidades, tipos o propiedades como datos de entrada en el proceso de edición. Por ejemplo, para una regla en la que la entidad

SensorTemperaturaOficina notifica el evento *TemperaturaCambiada*, si se desea comparar la *TemperaturaNueva* notificada por el *SensorTemperaturaOficina* con la propiedad *TemperaturaActual* de la entidad *ACOficina*, se debe hacer uso de la colección de entidades ajenas para acceder a esta última propiedad *TemperaturaActual*, ya que *ACOficina* no es ni el origen ni el destinatario de la regla.

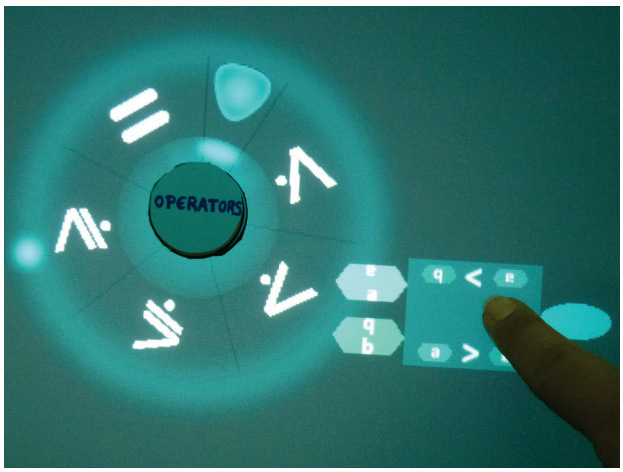


Figura 20. Selección y extracción de un operador al área de edición.

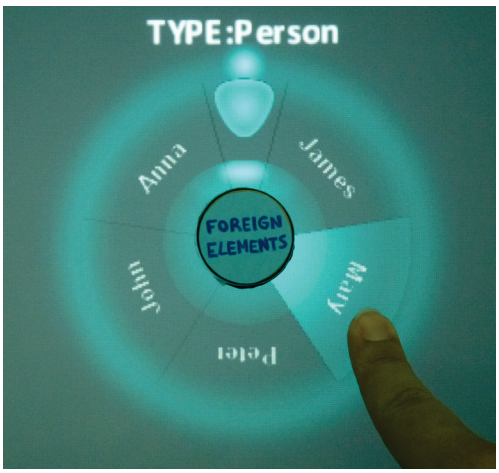


Figura 21. Selección de un elemento de la colección.

Todas las colecciones de las que se hace uso en el editor son accesibles mediante el empleo de tangibles asociados a las mismas. Los usuarios pueden tomar el tangible correspondiente, posicionarlo sobre la superficie, y de este modo la colección asociada se mostrará para su exploración. La rotación del tangible mientras la colección es visible en la pantalla permite mostrar más elementos de la colección que quedaban ocultos hasta el momento, tal y como sucede con los controles tradicionales de *scroll*, pero en este caso la rotación tiene lugar en 360 grados. Los tangibles pueden desplazarse a lo largo y ancho del área de edición para posicionar las colecciones en distintos lugares de la pantalla en función del espacio disponible o de las preferencias del usuario en cada momento. Cuando el usuario retira el tangible del área de edición, la colección asociada deja de ser visible. El estado de navegación de una colección no se guarda si el usuario levanta el tangible de la mesa. Las únicas colecciones que permanecen tal y como el usuario las dejó son las correspondientes al origen, evento y destinatario, una vez que se hayan establecido los mismos.

La navegación por los niveles de una colección se realiza pulsando sobre el elemento que se desea explorar con el dedo, movimiento conocido como *tap* (ver Figura 21), mientras que para incorporar un elemento a un proceso de datos se permite tanto el *tapping* como el arrastre del elemento al área de edición o *dragging* (ver Figura 20).

Por otro lado, para hacer más fácil la distinción de elementos durante la edición de la regla, las colecciones del origen, destinatario y evento tienen asociados códigos de colores, de modo que sus propiedades o atributos toman el color de la

colección correspondiente. En la Figura 22 se muestra un atributo de evento, del mismo color que la colección de eventos accesible con el tangible correspondiente.

Figura 22. Un evento y su atributo, en color naranja.

4.3.2. Vistas

El proceso de edición de una regla ha sido dividido en cuatro vistas principales por motivos de usabilidad. Esta distribución en vistas, que ya se empleó en el prototipo y resultó cómoda y satisfactoria, ofrece mayor espacio de trabajo para cada proceso de datos a editar, y permite focalizar la atención del usuario en una única expresión simultáneamente, reduciendo los errores cometidos.

La vista inicial de la edición de una regla se muestra en la Figura 23, y corresponde a la selección de las poblaciones origen y destinatario, así como del evento que activa la regla. Se trata de una primera toma de contacto del usuario con la regla a construir, de modo que esta vista constituye una especie de guía u orientación para concentrar al usuario en los elementos principales de la regla sobre los que versará el comportamiento a editar. En esta vista se muestran tres áreas coloreadas que parpadean, tratando de alentar al usuario a posicionar los tangibles que tiene disponibles del mismo color sobre su correspondiente área (amarillo para el origen, naranja para el evento y morado para el destinatario). Cuando se posiciona un tangible en su área asociada se muestra la colección correspondiente y es posible iniciar la exploración. Cabe destacar que no es posible comenzar la exploración de la colección de eventos hasta que la población origen ha sido seleccionada, debido a las restricciones impuestas en cuanto al tipo de eventos que puede lanzar cada población.

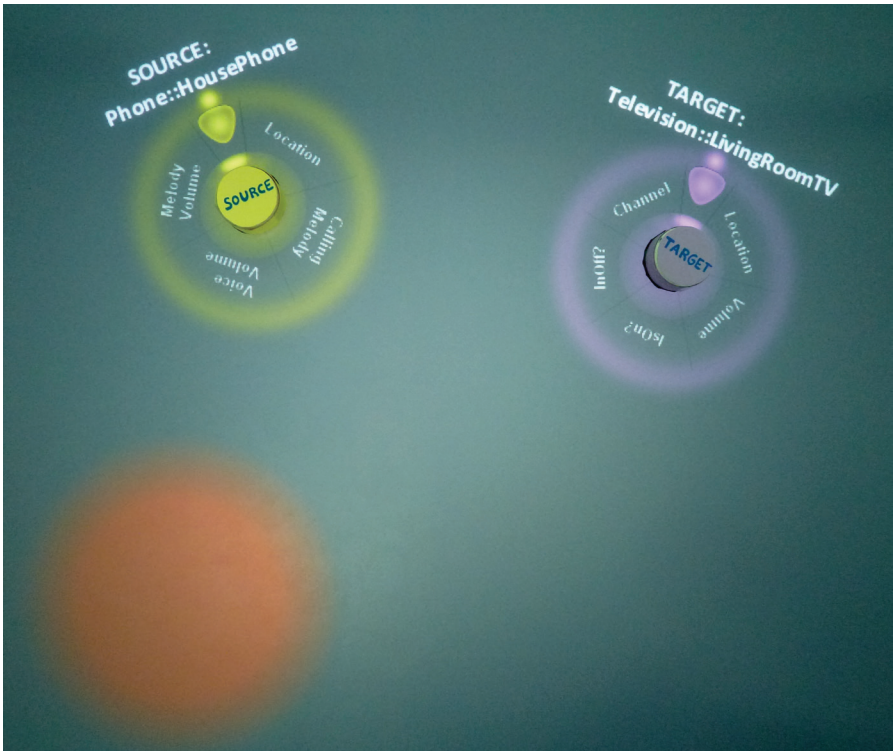


Figura 23. Vista inicial de edición de una regla.

Una vez que los tres elementos principales de la regla han sido seleccionados, los usuarios pueden escoger entre tres vistas diferentes: condición, filtrado y operación. Cada una de estas vistas corresponde con la edición de uno de los procesos de datos del mismo nombre que componen la regla. En ocasiones puede no ser necesario navegar por las tres vistas, ya que existen reglas en las cuales la condición o el filtrado son inexistentes, de modo que corresponde al usuario decidir en qué vista debe trabajar en cada caso.

4.3.3. Edición de flujos de datos

Cuando un usuario selecciona una de las vistas para editar su proceso de datos asociado, esta tarea involucra la incorporación de elementos al área de edición y el establecimiento de conexiones de flujos de datos entre dichos elementos para la composición de expresiones. Los elementos capaces de ser conectados mediante flujos de datos se denominan de manera genérica como nodos, puesto que la representación visual de un proceso de datos recuerda a una estructura en forma de árbol o grafo. Para las vistas de condición y filtrado inicialmente se muestra únicamente un nodo destino que representa el valor booleano resultante del proceso de datos. Por su parte, la vista de la operación inicialmente no muestra ningún nodo destino, ya que primero el usuario debe escoger o bien la propiedad o bien la acción de la población destino que se verá afectada en este proceso de datos. Cualquier expresión editada en un proceso de datos debe conectar el resultado de la expresión con el nodo destino correspondiente.

En cada proceso de datos los usuarios pueden hacer uso de propiedades, atributos, entidades, operadores o constantes numéricas para conformar una expresión. En la herramienta, las propiedades y atributos se seleccionan o arrastran desde las colecciones correspondientes hacia el área de edición, y lo mismo sucede con los operadores y entidades ajenas. Para añadir constantes numéricas a la expresión se ha diseñado un control especial con dicho propósito, mostrado en la Figura 24. Este control, al igual que las colecciones, está asociado a un tangible específico, y simula una regla numérica o *slider*. Haciendo uso de ambas manos, el usuario debe mantener pulsado el *slider* con el dedo, mientras que con la otra mano desplaza el tangible en busca del valor numérico deseado. Una vez se dispone de un nodo en el área de edición, los usuarios pueden desplazarlo a voluntad por toda la superficie simplemente tocando el nodo deseado con el dedo y desplazándolo sin levantar el dedo de la mesa.

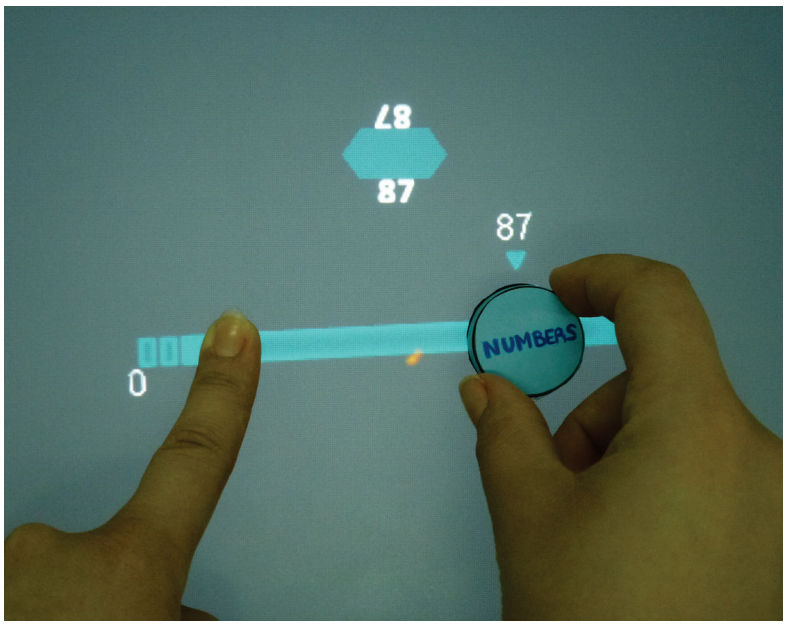


Figura 24. Control para la introducción de constantes numéricas en un proceso de datos.

La creación de flujos de datos entre elementos consiste en el simple dibujado de líneas entre los elementos a conectar. La Figura 25 refleja el proceso de creación de un flujo de datos entre dos nodos. Los flujos de datos únicamente pueden crearse desde los nodos productores de datos hacia los nodos consumidores de los mismos, es decir, siguiendo la dirección natural del flujo. El objetivo de esta unidireccionalidad no es otro que el de evitar errores fruto de la confusión de nodos productores con consumidores. La adecuación de esta decisión se podrá discutir tras observar los resultados del experimento presentado en la sección 0. De este modo, para crear un flujo de datos entre dos nodos el usuario debe tocar el nodo inicial de la conexión hasta que este se ilumine (lo cual sucede tras un retardo fijado a 1,5 segundos). En este momento el usuario puede mover su dedo por el área de edición, visualizando como este desplazamiento crea una línea recta de color azul claro entre el nodo origen y el contacto de su dedo en la superficie. Para crear correctamente una conexión, el usuario debe levantar el contacto sobre el nodo final del flujo de datos, de modo que la línea pasa a tomar otro color y

queda permanentemente conectada entre ambos nodos, aunque estos se desplacen. Si el usuario levanta el contacto antes de alcanzar algún nodo final, la línea se destruye y el flujo de datos no se crea.

En cualquier momento del proceso de edición, los usuarios pueden hacer uso de un tangible específico de borrado para eliminar cualquier nodo del área de edición, así como flujos de datos incorrectos, o incluso borrar el origen, evento o destinatario seleccionados para establecer uno diferente. Para ello, la interacción a realizar consiste en tocar el elemento a eliminar con el tangible de borrado. Cabe destacar que, en caso de eliminar la población origen o destino, o el evento que activa la regla, las propiedades o atributos relativos a los mismos que hayan sido empleadas en la regla se eliminan automáticamente de la misma. A su vez, si existe un flujo de datos entre dos nodos, dicho flujo de datos desaparecerá al borrar cualquiera de los dos nodos implicados en la conexión.

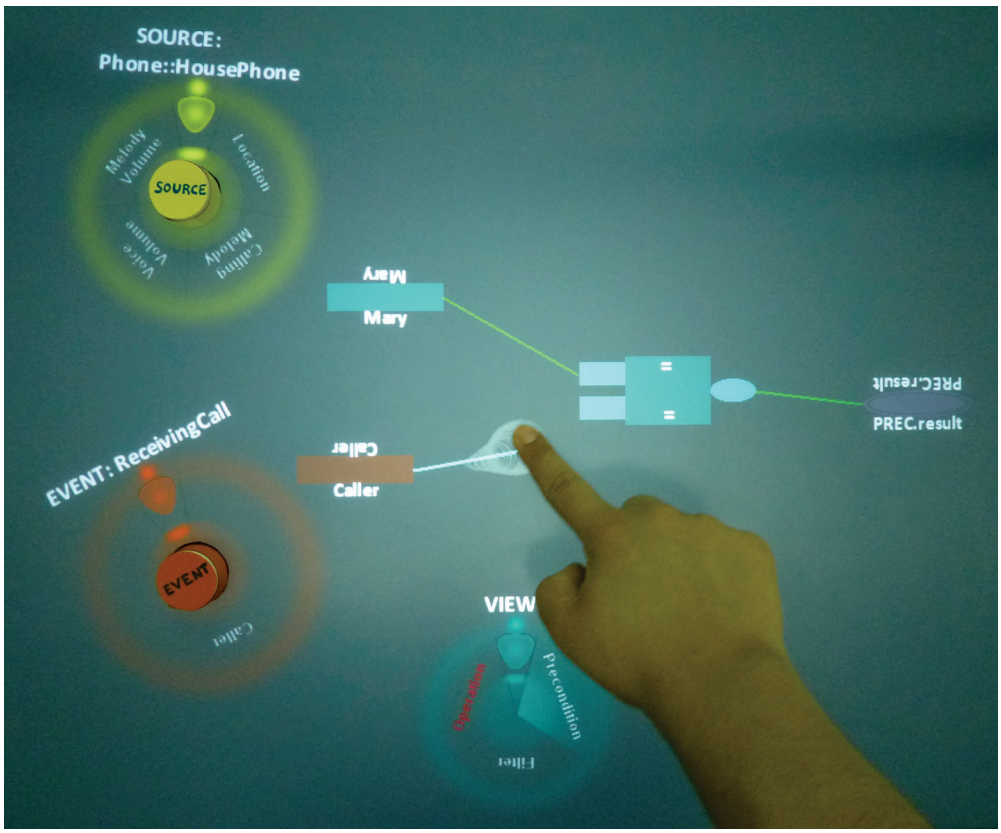


Figura 25. Conexión de un flujo de datos en la condición de una regla.

Finalmente, una característica altamente importante a tener en consideración en la herramienta es proveer retroalimentación o *feedback* visual para informar a los usuarios de cualquier error cometido durante la edición de un proceso de datos. Esta retroalimentación debe ser sencilla de comprender, a la vez que debe indicar concretamente dónde está el error para que el usuario pueda encontrar rápidamente maneras de solucionarlo. Se ha tratado de implementar esta característica de la manera más interactiva posible, de modo que la mayoría de errores se muestran mientras el usuario está realizando la edición. Por ejemplo, la creación de un flujo de datos debe darse entre dos elementos cuyos tipos sean

compatibles, ya que no sería correcto comparar dos propiedades numéricas con un operador booleano, etc. Cuando un flujo de datos conecta correctamente dos elementos del mismo tipo, la línea asociada toma el color verde, mientras que si los elementos son de distinto tipo y por tanto el usuario ha cometido un error, la línea asociada se vuelve roja. Tanto la población origen como la población destino, así como el evento, deben estar presentes en toda regla, de modo que si el usuario se olvida de establecerlos o los borra por equivocación, la colección asociada emite un parpadeo intermitente para atraer la atención del usuario. Otra de las informaciones interactivas que se reportan consiste en establecer en color rojo un nodo destino que todavía no tienen ningún flujo de datos conectado, puesto que es necesario que, si existe el proceso de datos, su resultado se asigne a alguna variable.

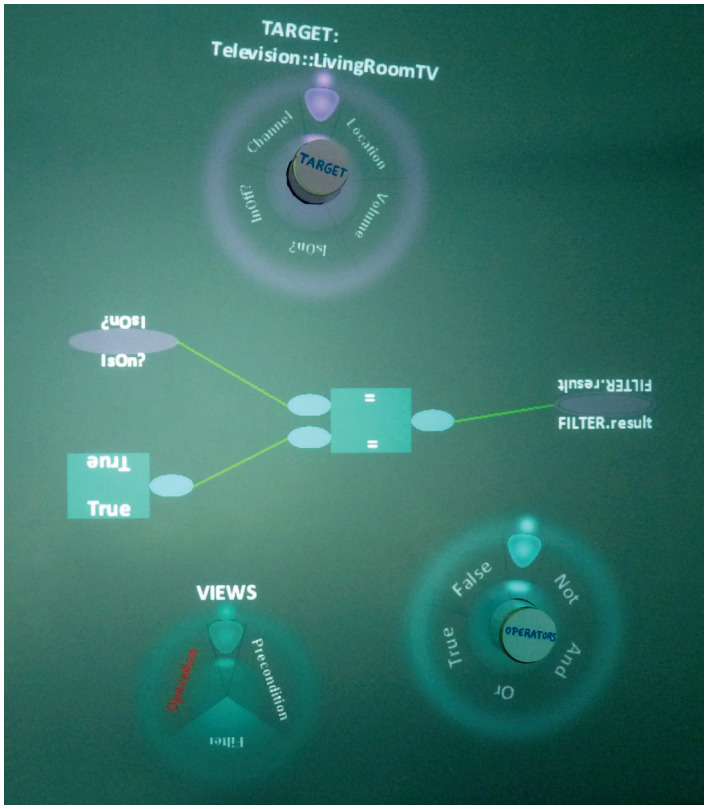


Figura 26. Edición del proceso de datos correspondiente al filtro de la regla.

Sin embargo, hay comprobaciones que se han considerado como costosas de realizar en tiempo de edición debido a la rapidez de las interacciones de los usuarios, y por tanto estas comprobaciones se realizan al finalizar la edición del proceso de datos en cuestión. Es el usuario quien, o bien pulsando sobre la vista a comprobar, o bien cambiando de vista, activa la batería de comprobaciones conocidas como *off-line*. Dichas comprobaciones *off-line* incluyen la detección de ciclos o la existencia de operadores para los que falta algún operando, en cuyo caso se indica en rojo el operando faltante. Sería necesario llevar a cabo pruebas de rendimiento para evaluar la viabilidad de incorporar estas comprobaciones de manera interactiva durante la edición. En cualquier caso, el menú de vistas indica en color rojo si se han detectado errores en una determinada vista, tal y como

aparece en la Figura 26. La Figura 27 muestra un proceso de datos completo y correctamente editado.

En resumen, el proceso de edición de una regla de comportamiento conlleva los siguientes pasos: en primer lugar, se deben seleccionar el origen, el destinatario y el evento de la regla en la vista inicial de edición, situando los tangibles correspondientes sobre las áreas coloreadas, y explorando las colecciones para buscar la entidad o evento deseado (Figura 23). Una vez establecidos estos tres ítems, el usuario puede escoger qué vista de la regla desea editar: condición (Figura 25), filtrado (Figura 26) u operación (Figura 27). El proceso lógico sería seleccionar en el menú de vistas la correspondiente a la condición de la regla, de modo que se muestra en la pantalla el área de edición vacía y un nodo booleano que representa el resultado de la condición. En este momento, el usuario debe editar el proceso de datos, formando la expresión correspondiente de manera visual, arrastrando (Figura 20), añadiendo (Figura 24) y conectando (Figura 25) los elementos necesarios en el área de edición. Al finalizar la condición, se pasaría a editar el filtrado siguiendo el mismo procedimiento. Tras el filtrado, el usuario pasaría a la vista de la operación, donde debe primero seleccionar de un menú circular como los empleados hasta ahora la propiedad o acción a editar. Una vez seleccionado esto, se puede editar el proceso de datos normalmente. Finalmente, las ayudas visuales permitirían detectar si existe algún error, y cuando la regla se encuentre correctamente editada el proceso de edición estará completo.

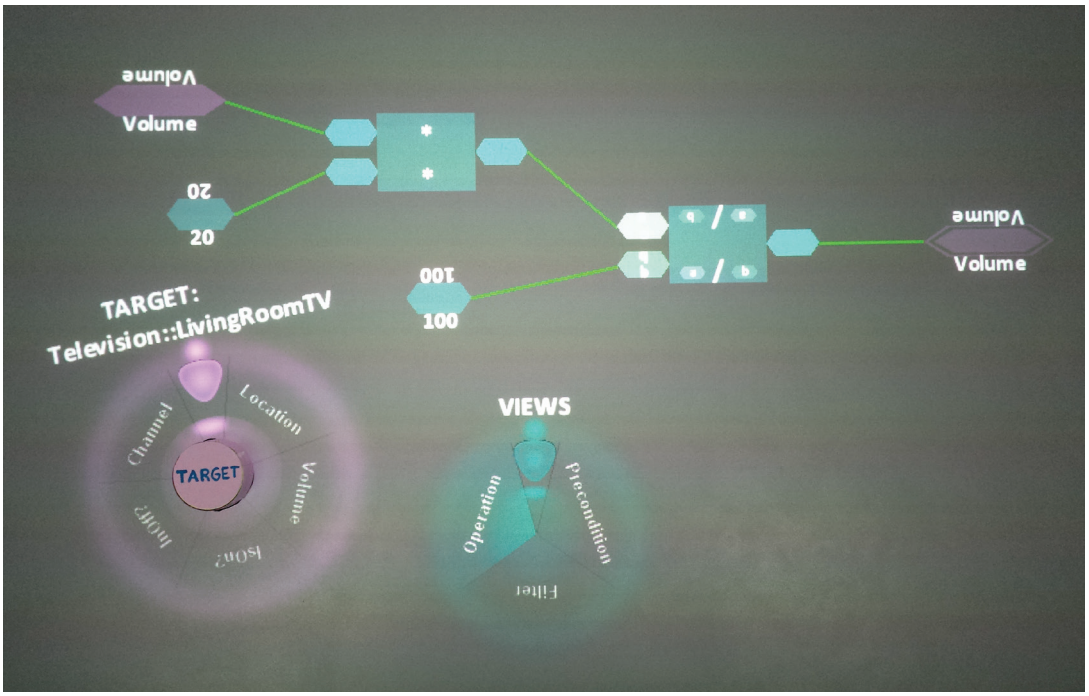


Figura 27. Proceso de datos de la operación de una regla completo y correcto.

5. Evaluación experimental

La siguiente sección presenta los resultados de dos evaluaciones llevadas a cabo sobre el presente trabajo. En primer lugar, en la sección 5.1 se resumen los resultados de distintas evaluaciones realizadas sobre el modelo de reglas y lenguaje visual propuestos en la sección 3, en relación a la comprensibilidad del lenguaje así como de la validación del mismo tras su integración en un sistema reactivo. En segundo lugar, la sección 5.2 detalla el experimento llevado a cabo para evaluar el proceso de edición de reglas de comportamiento como las propuestas empleando la herramienta descrita en la sección 4. En este caso, se explica el procedimiento empleado, qué aspectos de usabilidad han sido evaluados, se muestran los resultados obtenidos así como un análisis y discusión sobre los mismos.

5.1. Evaluación preliminar del modelo de reglas

En primer lugar se llevó a cabo una evaluación del lenguaje visual para la definición de expresiones propuesto, a fin de comprobar que resultaba fácil de comprender y utilizar, y así asegurar su efectividad en la posterior integración de este lenguaje en un editor de reglas gráfico como el que se presenta en este trabajo. Para ello, se condujo un experimento, tomando como participantes a 36 estudiantes de secundaria (25 chicos, 11 chicas) con una media de edad de 16 años. El experimento se llevó a cabo en el contexto de una asignatura de informática que los estudiantes se encontraban cursando, y ninguno de ellos reportó haber programado previamente al experimento. El profesor de la asignatura introdujo los conceptos necesarios para la definición de expresiones, tales como variables, operadores, asignaciones, etc. Finalmente se presentó a estos estudiantes tanto el lenguaje visual para definir expresiones basadas en flujos de datos, como un lenguaje textual equivalente, más familiar para ellos debido al uso frecuente de este tipo de nomenclatura en asignaturas como Matemáticas o Física. Para la evaluación, se les solicitó que realizaran cuatro tipos de ejercicios: cálculo, comparación, modificación y escritura de expresiones. Cada tipo de ejercicio debía realizarse en ocasiones empleando el lenguaje visual y en otras ocasiones el lenguaje textual, de manera proporcional. En cuanto a la corrección de las tareas propuestas, los ejercicios sobre cálculo y modificación de expresiones obtuvieron significativamente más respuestas correctas empleando el lenguaje visual, mientras que los ejercicios para la escritura y comparación de expresiones obtuvieron mejores resultados empleando el lenguaje textual, esta vez sin encontrarse diferencias significativas entre ambos tipos de lenguajes. También se pidió a los participantes que respondiesen un cuestionario sobre las dificultades percibidas en cada tarea, y sobre sus preferencias entre el lenguaje textual y el visual en distintas situaciones, así como la facilidad de uso y la comprensión de los mismos. De estos cuestionarios pudo observarse que los ejercicios considerados como más complejos fueron los de escritura y cálculo de expresiones, en ambos casos percibiéndose más dificultad con el lenguaje textual que con el visual.

Además, el lenguaje visual basado en flujos de datos resultó ser más fácil de emplear, de entender y de aprender, así como más útil a la hora de calcular expresiones. Más información sobre los resultados de este experimento puede encontrarse en *(Catalá, 2012)(Catalá, Pons, et al., 2013)*. Los positivos resultados de este estudio sugieren que la utilización del lenguaje visual propuesto facilitaría la definición de reglas de comportamiento de alta expresividad.

Para poder permitir la puesta en marcha y la evolución de los ecosistemas, se implementó un sistema de procesamiento de eventos que permitiese activar e instanciar reglas reactivas definidas en base al modelo ECA propuesto, así como una gramática que permitiese definir las ontologías necesarias para su carga e instanciación en el sistema. Se llevaron a cabo distintas pruebas para determinar la escalabilidad del motor de gestión y activación de reglas y su potencial uso en entornos reales con un elevado número de dispositivos, eventos y reglas, así como determinar la complejidad potencial alcanzable por las reglas en cuanto al número de operadores que incluyan. También se valoró el rendimiento del sistema paralelizando el procesamiento de eventos, de modo que en un futuro el sistema de gestión de reglas pudiese desplegarse sobre distintos nodos computacionales en caso de ecosistemas altamente complejos. En todas las dimensiones evaluadas, el sistema resultó ser altamente escalable y funcional. Resultados más detallados pueden verse en *(Catalá, 2012)(Catalá, Pons, et al., 2013)(Pons, 2012)*.

Finalmente, en el contexto del grupo de investigación ISSI se ha llevado a cabo el desarrollo de una plataforma basada en superficies interactivas para el fomento de la creatividad *(Catalá, 2012)(Catalá, García, Pons, et al., 2012)(Catalá, Jaen, et al., 2013)*. Esta plataforma permite la creación y simulación de juegos reactivos virtuales, permitiendo a los usuarios crear desde cero las entidades virtuales que formarán parte del ecosistema, dándoles apariencia visual y propiedades físicas. Con estas entidades creadas, es posible definir las reglas de comportamiento reactivas que permitirán su simulación, para lo cual se empleó el modelo de reglas propuesto en esta sección. Posteriormente, a partir del motor de procesamiento de eventos también comentado, y de un motor físico para la visualización y simulación de entidades reactivas, es posible llevar a cabo la coordinación de todos los elementos, dando vida a las entidades virtuales, interactuando con las mismas, y visualizando el resultado de las interacciones del usuario. La funcionalidad completa del middleware implementado se probó mediante el desarrollo de dos juegos reactivos conocidos, Pong[3] y Asteroids[4].

5.2. Evaluación del proceso de edición de reglas

Se ha llevado a cabo un experimento para evaluar la adecuación y usabilidad del editor de reglas desarrollado, presentado en la sección 4. Este experimento se ha enmarcado en el contexto de definición de reglas reactivas para una casa inteligente, comparando el desempeño de dos grupos de usuarios con distintos conocimientos sobre programación a la hora de especificar reglas de

[3] http://en.wikipedia.org/wiki/Pong
[4] http://en.wikipedia.org/wiki/Asteroids_%28video_game%29

comportamiento en dicho entorno. El objetivo del experimento consiste en determinar si los usuarios sin experiencia previa programando encuentran dificultades al enfrentarse a una tarea compleja como la especificación de comportamiento, empleando el modelo de reglas propuesto en la sección 3 y la herramienta para superficies interactivas basada en dicho modelo presentada en la sección 4. Los resultados derivados de este experimento permitirán determinar, además, la facilidad de uso con la que los usuarios perciben la interfaz propuesta, y la adecuación de los mecanismos de interacción multipunto y basados en tangibles empleados. La evaluación de estos aspectos, bien mediante análisis estadísticos o bien a partir de las experiencias reportadas por los usuarios, permitirá establecer si el proceso de especificación de comportamiento se ha visto afectado de alguna manera por dificultades a nivel de interacción o bien por problemas relacionados con la comprensibilidad del modelo conceptual de reglas subyacente.

A continuación se describe la evaluación de la herramienta de edición en base a los resultados obtenidos tras el experimento con usuarios programadores y no programadores. En primer lugar se establecen los aspectos a tener en cuenta para llevar a cabo la evaluación. A continuación se detallan los resultados estadísticos obtenidos a partir de los datos recuperados de los registros de interacción de los usuarios con el sistema, para cada uno de los aspectos evaluados, determinando si existen diferencias significativas entre los dos grupos de usuarios. Finalmente, se realiza un análisis cualitativo de los resultados obtenidos, y se tratará de determinar si la herramienta ha resultado ser un mecanismo efectivo para soportar la especificación de comportamiento en entornos inteligentes y reactivos.

5.2.1. Participantes

En el estudio han participado 16 voluntarios, 11 hombres y 5 mujeres, de los cuales 8 tenían experiencia previa programando (P), puesto que o bien se trataba de estudiantes de Ingeniería Informática, o bien de profesionales del sector. Los otros 8 sujetos no tenían conocimientos previos de programación (NP), ya que provienen de sectores no relacionados con el ámbito de la computación. Entre los no programadores se encontraban 1 estudiante de bachillerato, 4 estudiantes de grado, 1 intérprete con estudios de máster, 1 empresario también con estudios de máster, y 1 diplomado en fisioterapia. Las edades de los sujetos programadores varían entre 22 y 36 años (Media (M) = 27,75, Desviación Estándar (DE) = 5,676). Para los no programadores, el rango de edad varía entre los 17 y los 37 años (M = 26, DE = 7,874).

Los usuarios rellenaron un cuestionario previo indicando su frecuencia de uso de las tecnologías relacionadas con el experimento. Todos los sujetos programadores indicaron que empleaban tanto ordenadores personales como dispositivos táctiles a diario. Por su parte, siete de los no programadores indicaron emplear ordenadores a diario, y únicamente un no programador reportó utilizarlos casi a diario. En cuanto al uso de dispositivos táctiles, cinco de los no programadores los emplean a diario, dos de ellos indicaron su uso casi a diario, y únicamente un no programador indicó que rara vez utilizaba este tipo de dispositivos. Cuando se les preguntó por el uso de superficies interactivas,

únicamente uno de los participantes en el estudio, un programador, indicó haber empleado este tipo de tecnología muy ocasionalmente, debido a que dicho usuario había participado en experimentos previos del grupo de investigación. El resto de participantes no había tenido contacto previamente con superficies interactivas como la *Microsoft PixelSense*.

En cuanto a la dominancia manual, uno de los sujetos no programadores reportó ser ambidiestro, mientras que el resto de sujetos de ambos grupos indicaron ser diestros. Ningún participante indicó tener problemas de visión.

5.2.2. Metodología

Los participantes en el experimento debían utilizar la herramienta de edición presentada en la sección 4 para llevar a cabo una serie de tareas. El laboratorio donde se llevaron a cabo los experimentos contaba con dos unidades de la *Microsoft PixelSense*, de modo que únicamente dos sujetos podían estar trabajando a la vez en el laboratorio, uno en cada unidad, separados por una pantalla plegable. El primer paso consistió en presentar a los usuarios los conceptos necesarios sobre reglas reactivas y definición de comportamiento de manera incremental y ejemplificada, utilizando una ontología para el contexto de una casa inteligente. La ontología completa puede consultarse en el Anexo A. Las explicaciones pertinentes comenzaron por mostrar los conceptos más básicos de una regla, tales como las entidades o los eventos, y para la comprensión y memorización de los mismos se ofrecía una representación visual más comprensible, como la mostrada en el Anexo A. A fin de evitar vocabulario de índole más técnica, algunos conceptos fueron renombrados durante todo el experimento para emplear nombres más comprensibles por usuarios sin conocimientos previos de programación. Por ejemplo, los tipos de entidades eran referidos como *categorías de elementos*, y el concepto de entidad se denominaba *elemento concreto*. El evento de la regla fue denominado durante el experimento como *suceso*, tanto en los ejercicios como en la interfaz de la herramienta. Tras cada nuevo concepto introducido se llevaron a cabo una serie de ejercicios de autoevaluación que permitían a los usuarios determinar si habían comprendido correctamente las explicaciones. Desde estos conceptos más elementales, pasando por la definición de expresiones de manera visual como las mostradas en la sección 3.3, finalmente los participantes acabaron construyendo reglas completas con distintos procesos de datos de manera visual, y realizaron ejercicios de escritura y lectura de reglas de comportamiento completas de distinta dificultad empleando papel y lápiz. Para la definición de reglas de comportamiento, así como para la lectura de las mismas, se emplearon las plantillas mostradas en el Anexo B y Anexo C.

Una vez los participantes hubieron asimilado la ontología y el nuevo lenguaje visual de reglas de comportamiento a manejar, se les introdujo al manejo de la herramienta de edición, relacionando los conceptos de una regla con los elementos y controles ofrecidos por el editor. Se guió a los usuarios en la edición de una regla de ejemplo completa empleando la herramienta, de modo que pudiesen familiarizarse con las técnicas de interacción a emplear. El proceso de introducción a la herramienta duró entre 10 y 15 minutos.

Tarea	Dificultad de la condición	Dificultad de la operación	Presencia de elementos ajenos
Tarea 1	Baja	Baja	No
Tarea 2	Alta	Baja	No
Tarea 3	Baja	Baja	Sí
Tarea 4	Alta	Baja	Sí
Tarea 5	Baja	Alta	No
Tarea 6	Alta	Alta	No
Tarea 7	Baja	Alta	Sí
Tarea 8	Alta	Alta	Sí

Tabla 1. Dificultad de las reglas atendiendo a tres factores.

Tras la fase de introducción, los participantes debían completar 8 tareas, cada una de las cuales consistente en la definición de una regla de comportamiento empleando el editor de reglas. Para cada tarea se entregaba a los participantes la especificación visual en papel de la regla a editar, empleando las plantillas y la notación visual que habían aprendido. La dificultad de las reglas a editar variaba entre tareas, y la secuencia en la cual se llevaban a cabo las mismas era diferente para cada participante, a fin de prevenir que los resultados del experimento se viesen afectados por factores de ordenación. Para evitar que este experimento se alargase demasiado en el tiempo, se realizó una simplificación de las reglas mostradas a los usuarios, tanto en la parte conceptual como en el manejo de la herramienta, de modo que no se introdujeron a los usuarios ni las poblaciones colectivas ni la vista de filtrado. Esta decisión permite que se hayan continuado ofreciendo todos los mecanismos de interacción disponibles y los elementos necesarios para obtener una elevada expresividad, pero evitando introducir demasiados conceptos para una primera aproximación a la tarea de definición de comportamiento. Todos los participantes completaron las 8 tareas de edición con la herramienta en menos de 45 minutos. En la Tabla 1 se muestra la dificultad relativa de cada una de las reglas propuestas atendiendo a tres aspectos: la dificultad de la condición, la dificultad de la operación, y la presencia de elementos ajenos en la regla (propiedades y entidades no relacionadas con las poblaciones origen y destino de la regla). La dificultad de la condición y/o la operación se ha clasificado como *Alta* o *Baja*, considerando de dificultad *Baja* aquellas expresiones que involucran una comparación simple o una asignación trivial que incluya un único operador, p. ej., *Volumen - 20*. Paralelamente, las condiciones y operaciones con dificultad *Alta* son aquellas que involucran expresiones compuestas, p. ej., *Volumen > 20 Y Volumen < 30*. Simplificando esta clasificación, se considera que una expresión conlleva una *Alta* dificultad si involucra tres o más operadores, y *Baja* en caso contrario. Con respecto a los elementos ajenos, estos pueden estar presentes o no en la regla, de modo que la presencia de un único elemento ajeno en una expresión hace que la dificultad de la misma aumente.

Las interacciones de los usuarios al utilizar la herramienta han sido registradas, de modo que esta información ha sido post-procesada para extraer datos relevantes para el estudio estadístico que se presenta en las secciones 5.2.4, 5.2.5, 5.2.6. Además, las sesiones con la herramienta de todos los participantes han sido

grabadas en vídeo. Este material audiovisual ha permitido determinar las causas de algunos de los resultados obtenidos tras el experimento, y posibilita realizar análisis más detallados en trabajos futuros a fin de detectar patrones de interacción de los usuarios. Por ejemplo, a la hora de añadir varios operadores a una expresión, sería interesante saber si los usuarios realizan todas las extracciones a la vez y posteriormente editan la expresión, o si bien únicamente extraen el operador con el que van a trabajar cada vez y más tarde vuelven a explorar la colección para extraer el siguiente. Este tipo de comportamientos permitirían adaptar la interfaz en función de las preferencias del usuario, realizar sugerencias, o incluso plantear nuevos controles más adecuados según el uso que hacen los usuarios de los mismos.

Al finalizar las 8 tareas de edición con la herramienta, los participantes rellenaron un cuestionario indicando su opinión sobre aspectos de usabilidad del editor y sobre la comprensibilidad de los mecanismos de interacción empleados. Los resultados de dichos cuestionarios se analizarán en la sección 5.2.7.

5.2.3. Dimensiones de evaluación

Con el fin de evaluar la capacidad del proceso de edición para soportar la producción de reglas de comportamiento por parte de ambos grupos de usuarios, programadores y no programadores, y atendiendo a los atributos de calidad en uso de un producto definidos en el estándar ISO/IEC 25010:2011[5] (revisión de ISO/IEC 9126-1:2001), han sido estudiadas las siguientes dimensiones del proceso:

- *Rendimiento del proceso*, entendido como la capacidad de la herramienta para permitir a los usuarios alcanzar sus objetivos de manera efectiva y productiva. Puede medirse en base a distintos factores, tales como el tiempo empleado en cada tarea, exploraciones insatisfactorias realizadas o acciones incorrectas que deben deshacerse.
- *Corrección*, entendido como el grado de exactitud de las reglas obtenidas tras el proceso de edición atendiendo a la especificación en papel ofrecida.
- *Gestión de la disposición de elementos del proceso*, entendido como la adecuación de la disposición de los elementos de la interfaz en la herramienta teniendo en cuenta la distribución espacial y los movimientos que los usuarios realizan para posicionarlos.

5.2.4. Rendimiento del proceso

La dificultad del proceso de edición, y por tanto, el nivel de experiencia requerido para llevar a cabo dicha edición, pueden ser medidos en base a varias métricas. Las métricas que se han considerado en el presente experimento para

[5] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=22749

determinar el rendimiento de los usuarios en cada una de las tareas de edición propuestas son las siguientes: el tiempo medio empleado por los usuarios en cada tarea a fin de producir la regla deseada, el número de exploraciones en las colecciones que no conducen al resultado esperado, la cantidad de veces que los usuarios han cometido errores al extraer elementos al área de edición y el número de veces que se ha utilizado el tangible de borrado para eliminar elementos erróneos de la expresión. Cuanto más altos sean los resultados de estas métricas, más complejo y difícil habrá resultado el proceso de edición. En caso de obtener altos valores en estas métricas, deberían reconsiderarse los mecanismos de interacción empleados, ofreciendo a los usuarios métodos de edición más intuitivos y flexibles. Si, por el contrario, los resultados obtenidos mantienen valores bajos para estas métricas, el proceso de edición se entiende que ha resultado sencillo y que los mecanismos de interacción y sus metáforas asociadas han sido adecuados.

En primer lugar, se ha tomado el tiempo empleado por cada usuario en cada una de las tareas. Además, dentro de cada tarea se ha determinado el tiempo empleado por los usuarios en cada una de las distintas vistas de la regla. Con ello se podrá determinar si algunas partes de la regla han requerido mayor esfuerzo cognitivo por parte de los participantes. La Tabla 2 muestra el tiempo medio empleado en cada tarea según el grupo de participantes considerado, así como la desviación estándar obtenida en cada caso. Para determinar si hay diferencias significativas

Tarea	t	Grados de libertad (df)	p-value	Grupo	Media de tiempo (seg)	Desviación estándar (seg)	Error estándar (seg)
Tarea 1	1,868	14	0,083	NP P	151,684 118,680	44,816 22,101	15,845 7,814
Tarea 2	1,352	14	0,198	NP P	296,895 221,816	106,101 115,806	37,513 40,944
Tarea 3	1,819	14	0,090	NP P	113,011 96,471	23,613 10,190	8,349 3,603
Tarea 4	0,723	14	0,481	NP P	169,783 159,118	35,758 21,456	12,642 7,586
Tarea 5	2,241	14	0,042	NP P	378,510 265,731	133,836 48,469	47,318 17,136
Tarea 6	0,757	14	0,462	NP P	413,874 373,056	135,834 69,346	48,024 24,517
Tarea 7	0,990	10,859	0,344	NP P	376,355 305,764	176,877 96,963	62,536 34,281
Tarea 8	2,512	8,668	0,034	NP P	360,418 287,866	77,145 26,825	27,275 9,484

Tabla 2. Estadísticos para el tiempo medio empleado en cada tarea según grupo de usuarios.

entre las medias de tiempo de ambos grupos de usuarios para una misma tarea, se han realizado los correspondientes t-Tests o test de Student para comparar medias, asumiendo como hipótesis nula la igualdad de las mismas. Los resultados estadísticos de los t-Test se muestran también en la Tabla 2. Puede observarse que, si bien el tiempo medio empleado por los no programadores en cualquier tarea siempre es superior al tiempo medio empleado por los programadores, únicamente aparecen diferencias significativas entre ambos grupos en las tareas 5 (t_{14} = 2,241, p = 0,042) y 8 (t_{14} = 2,512, p = 0,034). Al tratar de determinar la razón por la cual existen diferencias en estos dos casos se ha examinado con detalle el tiempo medio empleado en cada vista para las dos tareas conflictivas. En ambas tareas se observa que únicamente aparecen diferencias significativas en el tiempo medio empleado en la vista de la operación: p = 0,008 para la tarea 5 y p = 0,033 para la tarea 8. Analizando los procesos de datos de las operaciones de estas dos reglas se observa que en ambos se realiza una distribución un tanto especial de los operadores en forma de cascada y no como árbol, tal y como se muestra en la Figura 15-c. Esta disposición del proceso de datos, sumada a que ambas operaciones son consideradas de alta dificultad por tener tres o más operadores, pueden ser la causa de las diferencias significativas de tiempo observadas entre ambos grupos de usuario. No obstante, pese a estas diferencias, los participantes se han comportado de manera similar atendiendo a los tiempos medios registrados para cada tarea, y el tiempo empleado en las mismas no se considera excesivo en ningún caso.

La siguiente métrica en cuanto a rendimiento en el proceso de edición está relacionada con el manejo de colecciones empleando un novedoso control 360 grados que permite la visualización, posicionamiento y exploración de colecciones por medio de tangibles e interacciones con los dedos, tal y como se mostró en la sección 4.3.1. La exploración de estas colecciones no siempre conduce a los participantes a los resultados deseados, ya que en ocasiones no encuentran el elemento buscado y deben volver atrás en el proceso, o bien seleccionan por error un elemento incorrecto. De este modo, se denominan exploraciones indeseadas a aquellas exploraciones que no conducen ni a la selección de la población origen o destino, ni a la selección del evento, ni tampoco a la extracción de elementos al área de edición. Dichas exploraciones indeseadas fuerzan a los participantes o bien a emplear el tangible de borrado sobre la colección o bien a levantar el tangible asociado a la colección de la mesa y volver a posicionarlo para retornar al estado inicial de la colección. La Figura 28 muestra el número total de exploraciones indeseadas para cada tipo de colección de las disponibles en el editor. Se puede observar que la colección de operadores es la causante de la mayor parte de los problemas de exploración detectados, provocando el 66,67% de las exploraciones indeseadas de los no programadores y el 56,14% de las exploraciones indeseadas de los programadores. Esta colección muestra inicialmente un conjunto de categorías en las cuales se agrupan los operadores: operadores lógicos, operadores relacionales, operadores aritméticos y selectores. Cuando el usuario selecciona una de las categorías los operadores de la misma se muestran en la colección. Normalmente, los usuarios fallan al seleccionar la categoría en la que creen que se

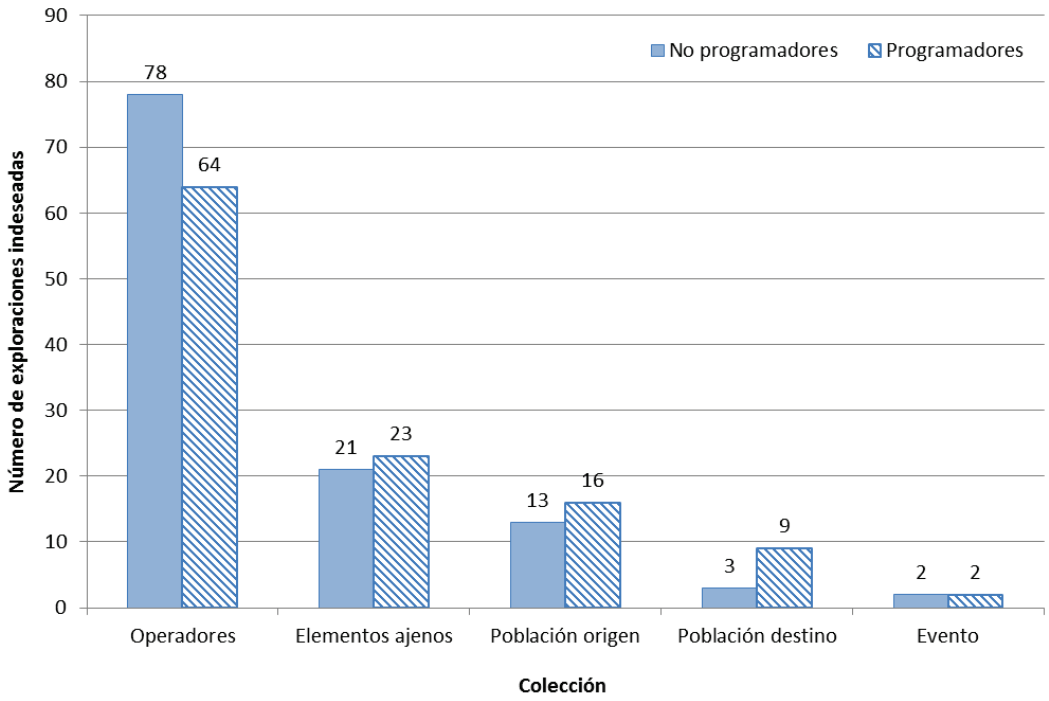


Figura 28. Número de exploraciones indeseadas según grupo de usuarios y tipo de colección.

encuentra el operador deseado, ya que el nombre de la categoría se muestra de manera textual y no hay modo de saber qué operadores contiene salvo que se exploren las categorías una por una. Es necesario plantear una agrupación más visual e intuitiva de los operadores en futuras versiones del editor para resolver las dificultades encontradas con esta colección.

Para determinar si existen diferencias significativas entre programadores y no programadores con respecto al número total de exploraciones indeseadas, se ha llevado a cabo el test de Mann-Whitney para cada tipo de colección. Los resultados de dicho test se muestran en la Tabla 3, e indican que no existen diferencias significativas entre ambos grupos de usuarios.

La tercera métrica considerada en esta dimensión corresponde al número de nodos que son extraídos al área de edición pero que no son necesarios para la

Colección	Mann-Whitney U	Z	p-value
Operadores	20,5	-1,218	0,223
Elementos ajenos	29,5	-0,272	0,786
Población origen	27,5	-0,485	0,628
Población destino	21,0	-1,272	0,203
Evento	29,0	-0,463	0,643

Tabla 3. Análisis estadístico de diferencias significativas para el número de exploraciones indeseadas según tipo de colección.

definición de la regla. Esta métrica indica si los usuarios son propensos a cometer errores cuando tienen que seleccionar las propiedades, operadores o demás elementos necesarios de las colecciones disponibles para editar un proceso de datos. La Tabla 4 resume el análisis estadístico derivado de los tests de Mann-Whitney ejecutados sobre el número de nodos no necesarios extraídos en cada tarea, y no se aprecian diferencias significativas entre los dos grupos de usuarios para ninguna de las tareas propuestas. Además, no se considera excesivo el número de nodos no necesarios extraídos por ninguno de los dos grupos de usuarios, ya que consiste en un 8,8% de los nodos totales extraídos en el caso de los programadores, y un 18,6% en el caso de los no programadores.

Otra de las métricas a considerar evalúa el número de elementos que se han eliminado del área de edición, ya se trate de nodos o de flujos de datos. En el

Tarea	Mann-Whitney U	Z	p-value
Tarea 1	23,5	-1,179	0,239
Tarea 2	28,0	-0,460	0,646
Tarea 3	20,0	-1,852	0,064
Tarea 4	19,0	-1,554	0,120
Tarea 5	18,5	-1,492	0,136
Tarea 6	27,0	-0,535	0,593
Tarea 7	19,5	-1,415	0,157
Tarea 8	25,5	-0,699	0,485

Tabla 4. Análisis estadístico de diferencias significativas para el número de nodos no necesarios extraídos en cada tarea.

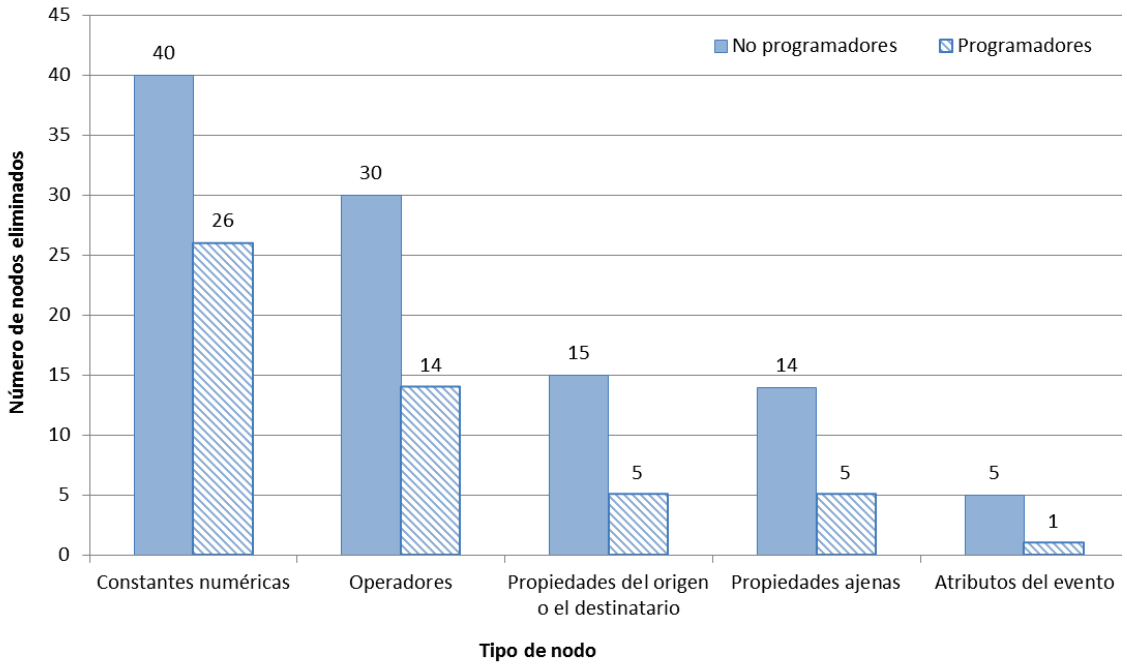


Figura 29. Número de nodos eliminados por grupo de usuario según tipo de nodo.

primer caso, resulta interesante determinar qué tipos de nodos son los que los usuarios han eliminado con mayor frecuencia empleando el tangible de borrado. Tal y como puede observarse en la Figura 29, las constantes numéricas han sido los nodos que han provocado más veces el uso del tangible de borrado, agrupando un 42,58% de las eliminaciones de todos los participantes. El control asociado a la inserción de constantes numéricas no resulta satisfactorio para muchos de los usuarios, hecho que puede deducirse tanto del análisis cuantitativo de eliminaciones como de los cuestionarios personales que se reportarán en la sección 5.2.7. El siguiente tipo de nodo que más problemas ha causado ha sido el tipo de los operadores, con un 28,39% del total de eliminaciones. El problema con los operadores erróneos viene derivado del uso de una representación en 360 grados. Determinados operadores, como son por ejemplo el "*mayor que*" y "*menor que*", tienen una representación simétrica invertida, lo cual no supone problemas cuando la orientación de la lectura está clara. Por el contrario, al haberse considerado la posibilidad de la edición de reglas colaborativa, la lectura puede darse en cualquier ángulo, y por tanto es difícil distinguir correctamente entre estos dos operadores, a pesar de las ayudas visuales ofrecidas (la parte inferior del operador se ha marcado con un punto blanco, como muestra la Figura 20).

En general se observa que los no programadores siempre realizan más borrados que los programadores, como cabía esperar. Sin embargo, los test de Mann-Whitney realizados sobre el número de nodos de cada tipo eliminados no detectan diferencias significativas entre programadores y no programadores para ningún tipo de nodo (ver Tabla 5). No se considera excesivo el total de borrados realizados

Tipo de nodo	**Mann-Whitney U**	**Z**	**p-value**
Constantes numéricas	23,5	-0,904	0,366
Operadores	16,0	-1,721	0,085
Propiedades del origen y el destinatario	15,0	-1,864	0,062
Propiedades ajenas	21,5	-1,188	0,235
Atributos del evento	23,5	-1,179	0,239

Tabla 5. Análisis estadístico del número de nodos de cada tipo eliminados mediante el tangible de borrado.

Tarea	**Mann-Whitney U**	**Z**	**p-value**
Tarea 1	32,0	0,0	1,0
Tarea 2	24,0	-1,461	0,144
Tarea 3	32,0	0,0	1,0
Tarea 4	28,0	-1,0	0,317
Tarea 5	24,0	-1,464	0,143
Tarea 6	20,0	-1,852	0,064
Tarea 7	24,0	-1,461	0,144
Tarea 8	28,0	-0,616	0,538

Tabla 6. Análisis estadístico del número de flujos de datos eliminados mediante el tangible de borrado en cada tarea

por ninguno de los dos grupos de usuarios, ya que los no programadores borraron un 16,02% de los nodos extraídos, mientras que los programadores borraron un 8,8%.

En cuanto al número de borrados de flujos de datos empleando el tangible de borrado, la Tabla 6 muestra el resultado de los tests de Mann-Whitney realizados para determinar si existen diferencias significativas a este respecto entre programadores y no programadores, determinándose que no existen dichas diferencias para ninguna de las tareas propuestas. Tampoco se considera excesivo el número de flujos de datos borrados por ninguno de los dos grupos de usuarios, ya que el porcentaje de flujos de datos borrados sobre el número de flujos de datos extraídos por programadores y no programadores es de 2,55% y 2,33%, respectivamente.

Tras los análisis estadísticos realizados, puede considerarse que los resultados obtenidos para estas métricas son aceptables, si bien hay algunos aspectos de usabilidad que deberían ser mejorados y que se discutirán más adelante.

5.2.5. Corrección de las reglas

Con el objetivo de determinar si las reglas definidas por los no programadores contienen significativamente más errores que aquellas definidas por los programadores, se ha llevado a cabo un análisis de las reglas editadas durante la sesión de experimentación con la herramienta. En este apartado se resume qué tipos de errores se han producido, cuáles de ellos son los más comunes, y se trata de dar explicación a la ocurrencia de dichos errores a fin de encontrar soluciones que permitan prevenir en futuras versiones del editor este tipo de fallos, mejorando por tanto el proceso de edición.

Considerando las 8 tareas asignadas a cada uno de los usuarios, los no programadores cometieron un total de 17 errores (M = 2,13, DE = 3,271, Mediana (Md) = 0,5) en contraste con los 7 errores cometidos por los programadores (M = 0,88, DE = 0,641, Md = 1,00). A pesar del elevado número de errores cometidos por los no programadores, los tests no paramétricos de Mann-Whitney muestran que no existen diferencias significativas entre ambos grupos de usuarios en cuanto al número de errores cometidos (U = 32; Z = 0,0; p = 1,00) debido a que muchos de los errores de los no programadores fueron cometidos únicamente por dos de ellos, tal y como se verá más adelante. Atendiendo a la distribución de errores según la vista de la regla en la que han sido cometidos, para los no programadores esta distribución consiste en 4 errores en la selección del evento, origen o destinatario, 3 errores en las condiciones y 10 errores en las operaciones. Los programadores no cometieron ningún error al establecer el evento, origen y destinatario, pero en este grupo de usuarios han sido detectados 6 errores en las condiciones y 1 error en una operación.

Los 4 errores cometidos en la vista inicial de la regla, es decir, en la selección del evento, origen o destinatario, han sido provocados por dos causas diferentes relacionadas con la interfaz y no con problemas conceptuales de los usuarios. En 3 ocasiones se produjo la incorrecta selección del origen o destinatario de la regla, ya

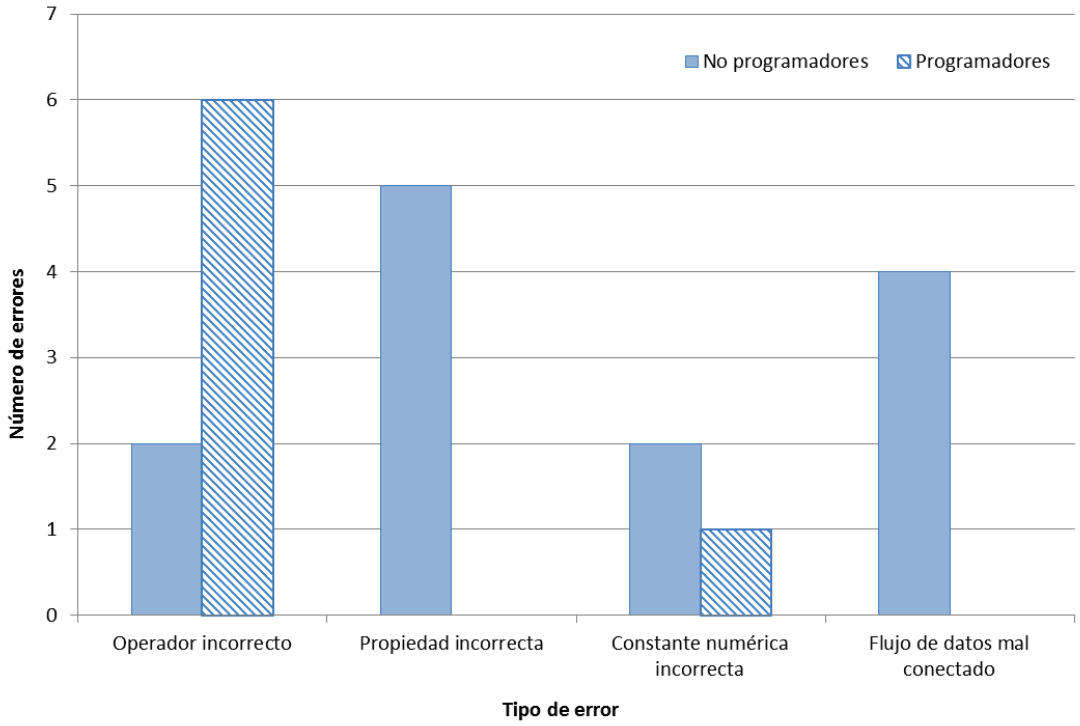


Figura 30. Número de errores de cada tipo según el grupo de usuarios considerado.

que el usuario seleccionó poblaciones cuyos nombres eran muy similares, de modo que el problema recae en la difícil legibilidad de las colecciones de poblaciones. Por otra parte, el error restante fue debido a que un usuario borró el evento de la regla y posteriormente olvidó volver a seleccionarlo, ya que mientras la colección no es visible no existe ningún elemento visual recordatorio para notificar esta inconsistencia.

Al tratar de clasificar los 20 errores cometidos en la edición de los procesos de datos de las reglas surgen cuatro tipologías de error: operadores incorrectos, propiedades incorrectas, constantes numéricas incorrectas y flujos de datos mal conectados. La Figura 30 muestra la distribución de estos cuatro tipos de errores en los dos grupos de usuarios. A continuación se detalla en qué consiste cada uno de ellos, indicando las posibles causas que han provocado la aparición de estos errores.

Los errores de *operador incorrecto* implican que el usuario ha seleccionado en la herramienta un operador diferente al indicado en la especificación en papel de la regla en la que está trabajando. Este tipo de error es el más común entre los programadores, y se ha observado que siempre que se produce la confusión, ésta se da entre los operadores "*mayor que*" y el "*menor que*". La confusión entre estos dos operadores, tal y como se explicó en el apartado anterior, se produce debido a la similitud de la representación visual de los mismos cuando se emplea en superficies 360 grados.

Cuando se produce un error consistente en una *propiedad incorrecta*, significa que el usuario ha extraído al área de edición una propiedad diferente a la

especificada en la regla provista. Tal y como muestran los vídeos de la sesión con la herramienta, estos errores han sucedido en los siguientes casos: en dos ocasiones, el participante no miró a la vista correcta de la especificación cuando estaba editando, de modo que empleó propiedades que aparecían en la condición mientras estaba editando la operación, puesto que los nombres de las mismas eran similares; en otras dos ocasiones, el participante seleccionó una propiedad con el mismo nombre que la mostrada en la especificación de la regla, pero perteneciente a otra población cuyo nombre era similar al de la población correcta (i.e.: *PersianaOficina.Nivel* y *PersianaCocina.Nivel*); y finalmente en una ocasión el error fue provocado por la incomprensión de la operación a ser editada. Estas evidencias sugieren que la representación textual de los elementos conduce a error, sobretodo trabajando con controles 360 grados. Para evitar confusiones entre elementos cuya representación textual sea similar, como en los casos descritos, sería necesario dotar a la herramienta de una representación más visual de determinados aspectos.

En lo que respecta a los errores debidos a *constantes numéricas incorrectas*, la principal causa se encuentra en el control que provee la interfaz para la introducción de estos valores en el proceso de datos. Tal y como se mostró en el apartado anterior, el control para introducir constantes numéricas provocaba que los usuarios fallaran al introducir el valor deseado, de modo que debían borrar el nodo recién creado, aumentando así el número de borrados realizados con nodos de este tipo. En este caso, los errores introducidos por el control de introducción de constantes numéricas van más lejos, provocando a los usuarios fallos de los que no llegan a ser conscientes. Debido a que en diversos procesos de datos se requiere la introducción de varios valores numéricos, por comodidad los usuarios suelen realizar estas interacciones de manera concatenada, insertando todas las constantes numéricas necesarias antes de dejar pasar a editar otro aspecto del proceso. Un uso incorrecto del control para la inserción de constantes numéricas ha conducido a los usuarios a modificar accidentalmente valores de inserciones previas sin que el participante fuese consciente, de modo que no llegaban a borrar el valor incorrecto para corregirlo, y la regla finalmente ha resultado ser errónea.

En último lugar, los errores debidos a los *flujos de datos mal conectados* consideran aquellas propiedades, constantes numéricas, etc., que no se conectan al nodo de entrada correcto. En ocasiones, esto es debido a la similitud entre dos nombres de propiedades, lo cual ha provocado que el usuario las confunda e intercambie sus conexiones. En otros casos, los procesos de datos cuya distribución de elementos seguía una ordenación en cascada (ver Figura 15-c) han sido fruto también de conexiones incorrectas.

En la Figura 31 puede observarse un histograma que representa el número máximo de errores cometidos por un único usuario en cada uno de los grupos de participantes. A este respecto, el número máximo de errores cometidos por un único programador durante la sesión completa, es decir, teniendo en cuenta la edición de las 8 tareas, es de dos errores. Únicamente un programador cometió dos errores, mientras que cinco de ellos cometieron un único error en toda la sesión, y solamente dos llevaron a cabo las 8 tareas sin ningún fallo. En cuanto a los no programadores, puede observarse que hay dos sujetos en este grupo que destacan debido al número máximo de errores cometidos, 9 y 5 respectivamente. Se ha analizado el tipo de errores cometidos para estos dos sujetos, a fin de detectar si éstos se deben a una incomprensión del modelo de reglas con el que se trabaja, o bien si es debido a aspectos relacionados con la interacción con la herramienta. Los 9 errores cometidos por uno de estos no programadores pueden ser clasificados en las siguientes categorías: en dos ocasiones el error fue debido a la confusión entre dos operadores, "*mayor que*" y "*menor que*", tal y como le sucedía a muchos programadores; en cuatro ocasiones se empleó una propiedad incorrecta; y en tres ocasiones el error se produjo al seleccionar una población origen o destino errónea. En general, los errores cometidos por este usuario están relacionados con aspectos de legibilidad del editor, ya que las confusiones se han provocado siempre entre elementos con nombres similares. En cuanto al no programador que cometió 5 errores, la distribución de los mismos queda como sigue: dos de los errores se deben a constantes numéricas incorrectas, mientras que los tres errores restantes se han producido debido a la conexión incorrecta de tres flujos de datos en la

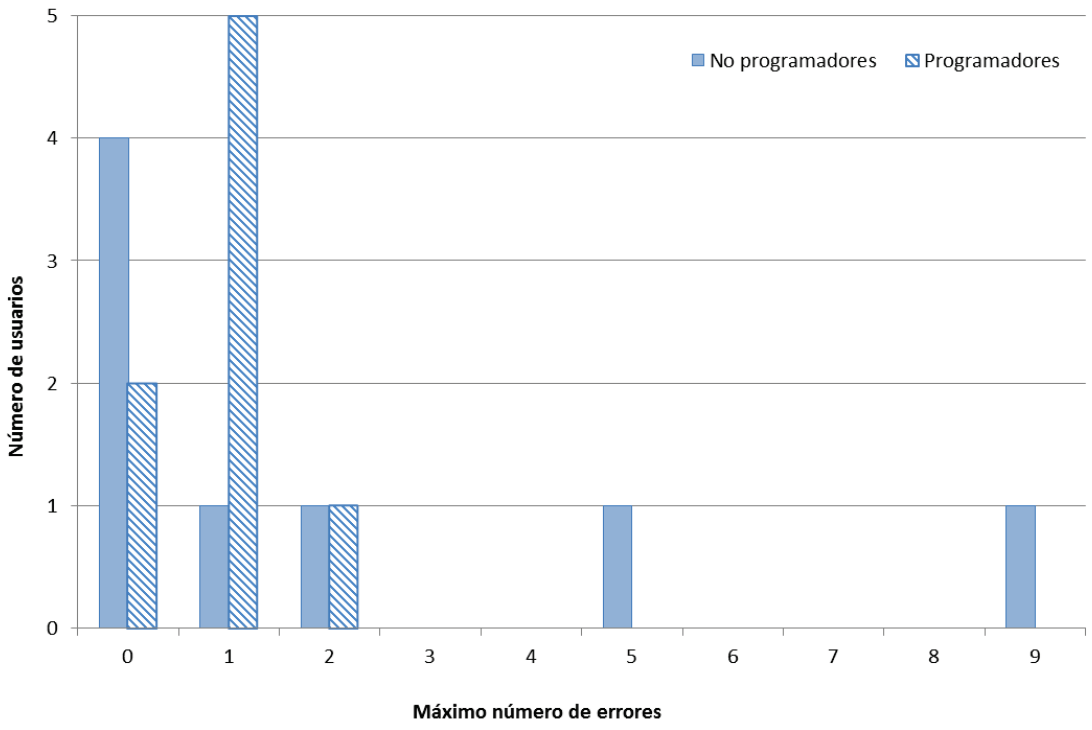


Figura 31. Histograma para el máximo número de errores cometido por cada usuario, según grupo de usuarios.

misma operación, y cabe destacar que se trataba de una operación en forma de cascada. Las causas que han provocado el elevado número de errores de este usuario pueden ser parcialmente explicadas por el breve período de tiempo que el sujeto empleó para realizar las distintas tareas del experimento, tal y como puede observarse en los vídeos y en los registros de interacción. De hecho, este usuario llegó a completar las 8 tareas del experimento en menos tiempo que algunos de los programadores, y por tanto resulta altamente probable que no dedicase el tiempo suficiente a leer y comprender bien la regla a editar, ni a comprobar tras la edición que el significado era el esperado.

Tras este análisis puede deducirse que los errores cometidos en la definición de las reglas no han recaído sobre problemas de comprensión conceptual del modelo, sino más bien sobre aspectos de la interfaz que han llevado a confusión.

5.2.6. Gestión de la disposición de elementos del proceso

El principal objeto de análisis en esta dimensión consiste en evaluar la adecuación de la distribución de los elementos gráficos de la interfaz a lo largo del área de edición. Para este análisis se tienen en cuenta aquellas interacciones que no afectan al significado de la regla pero que ayudan a los usuarios a sentirse más cómodos con la interfaz gráfica. Por tanto, se ha diferenciado entre dos tipos de acciones a ser estudiadas, relacionadas con el movimiento y *dragging* de los elementos por la superficie: movimiento de nodos y movimiento de colecciones. El análisis del movimiento de estos dos elementos de la interfaz resaltará si la disposición actual de los mismos se ha considerado problemática, inadecuada o particularmente molesta para los participantes. También se ha estudiado la disposición preferida de los usuarios para las distintas colecciones que ofrece la interfaz, de modo que, en caso de reconsiderar una redistribución de dichas colecciones, ésta sea lo más cercana posible a las necesidades reales de los usuarios mientras trabajan con la herramienta.

Cuando los usuarios editan un proceso de datos, extraen elementos al área de edición. Dichos elementos, denominados genéricamente nodos, pueden ser reubicados a lo largo y ancho de toda la superficie para ocupar el lugar que el usuario considere más pertinente en cada caso. El estudio sobre el número de movimientos realizados con los nodos pretende determinar si estas reubicaciones son o no frecuentes, y si alguno de los dos grupos de usuarios ha destinado más esfuerzos en llevar a cabo el posicionamiento de nodos durante la edición. En este caso, emplear como medida a comparar el número de movimientos realizados con los nodos en cada una de las tareas no permite distinguir si dichos movimientos son excesivos. Una primera aproximación para obtener una medida que permita relativizar si el número de movimientos es excesivo o no sería dividir el número de movimientos realizados en una tarea por el número de nodos requeridos para completar la misma. No obstante, no es una medida acertada, ya que los usuarios pueden haber cometido equivocaciones durante la edición y haber extraído nodos

que luego han borrado por ser incorrectos. Por tanto, se ha tomado como medida de comparación el *ratio de movimiento excesivo de nodos*, calculado como sigue:

$$RMEN = \frac{número\ de\ movimientos\ realizados\ con\ los\ nodos\ en\ una\ regla}{número\ de\ nodos\ extraídos\ en\ una\ regla}$$

Esta medida toma en consideración que aquellos usuarios que hayan extraído más nodos al área de edición, incluso aquellos que se hayan extraído por error o aquellos que no se hayan utilizado finalmente en la edición, tienen más probabilidad de realizar más movimientos con los mismos. De este modo, asumiendo que todos los nodos se han movido por igual, un $RMEN < 1$ indicaría que el usuario en cuestión no ha llegado a reubicar todos los nodos que ha extraído. Por el contrario, un $RMEN > 1$ indica que el participante ha realizado como mínimo una reubicación de cada nodo. La Figura 32 muestra la media del RMEN para cada una de las tareas, diferenciando entre ambos grupos de usuarios. Puede observarse que prácticamente en todas las tareas, los no programadores han realizado mayor número de reubicaciones de los nodos, y que en ningún caso se realizaron más de dos reubicaciones por nodo, lo cual supone una medida aceptable en términos de comodidad y usabilidad en el proceso de edición. Por otro lado, la Tabla 7 muestra los resultados obtenidos tras realizar t-Test sobre los RMEN de ambos grupos de usuarios para cada una de las tareas. Únicamente aparecen diferencias significativas en las tareas 2 (p = 0,045) y 4 (p = 0,013), en las cuales los no programadores han movido más nodos. Un análisis más profundo de los resultados ha permitido detectar que en cada una de estas tareas hubo dos usuarios no programadores que realizaron un elevado número de movimientos de los nodos. El análisis de los vídeos grabados durante las sesiones refleja que este comportamiento se debe a que dichos usuarios realizaron distintas redistribuciones de los elementos en estas tareas. Estas redistribuciones

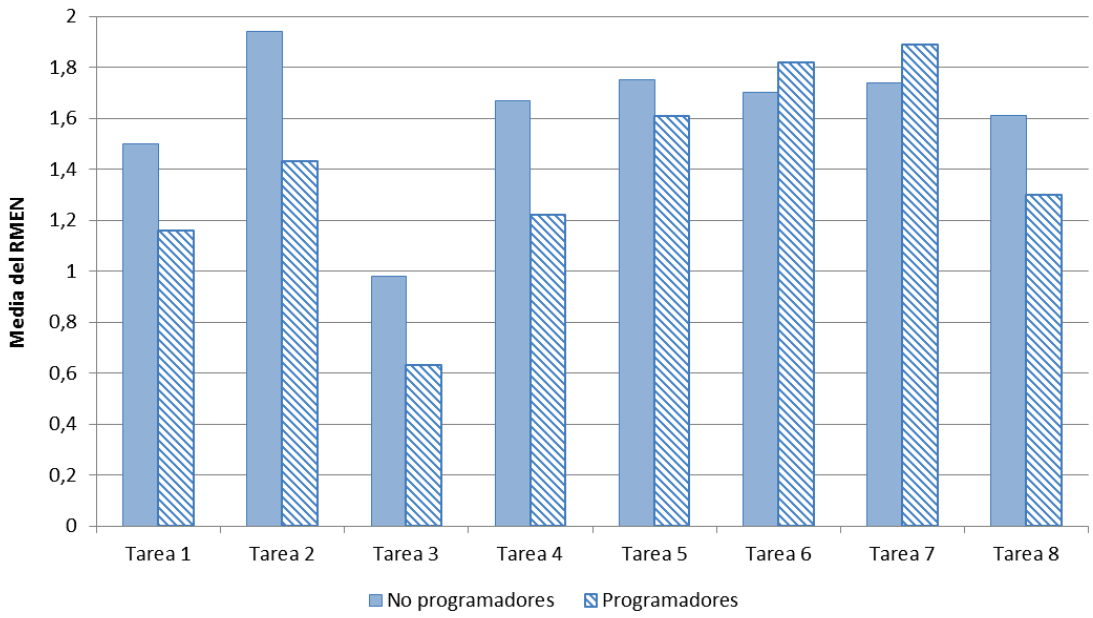


Figura 32. Media del RMEN para cada tarea según grupo de usuarios.

Tarea	t	Grados de libertad (df)	p-value
Tarea 1	1,334	14	0,203
Tarea 2	2,196	14	0,045
Tarea 3	1,288	14	0,219
Tarea 4	2,826	14	0,013
Tarea 5	0,461	14	0,652
Tarea 6	-0,479	14	0,639
Tarea 7	-0,516	14	0,614
Tarea 8	2,016	14	0,063

Tabla 7. Análisis estadístico para el RMEN en cada una de las tareas.

sucedieron después de que el usuario detectase que había cometido un error, eliminando los elementos necesarios y reubicando los restantes, así como añadiendo nuevos elementos al proceso.

A través de estos vídeos se han podido determinar las causas que, en general, conducen a los usuarios a redistribuir los nodos por el área de edición. Tal y como se comentó en la sección 4.3.3, las propiedades, los atributos del evento y los operadores pueden ser añadidos al área de edición de dos modos: pulsando con el dedo sobre el elemento (*tapping*) o arrastrando el elemento desde la colección hasta el área de edición (*dragging*). De todas las extracciones de elementos que se han realizado en las 8 tareas considerando a los usuarios de ambos grupos, el 64,18% de las veces dicha extracción se llevó a cabo mediante *tapping*, mientras que el 35,82% de las veces el elemento fue extraído mediante *dragging*. La extracción de elementos mediante *dragging* involucra un movimiento del nodo, que dura desde el momento en que se selecciona en la colección hasta que se posiciona en la ubicación deseada del área de edición, sin levantar el dedo de la pantalla. En cambio, al extraer un elemento de una colección mediante *tapping*, el nodo asociado a dicho elemento se ubica en la primera región libre más cercana a la colección. La mayoría de veces, esta posición no es la más adecuada o lógica, ya que no es próxima al resto del proceso de datos, de modo que el usuario tiene que, forzosamente, desplazar el nodo por la pantalla para llevarlo a una posición más conveniente. Es probable que el número de movimientos efectuados sobre los nodos disminuyese si el algoritmo de posicionamiento de nodos tuviese en cuenta otros factores, tales como la estructura de la regla que está siendo construida, de modo que los nodos extraídos mediante *tapping* ocupasen posiciones próximas a operadores cuyos argumentos de entrada estén sin conectar y tengan tipos compatibles.

Teniendo en cuenta el número de veces que se han movido las colecciones sobre la superficie interactiva, se han realizado análisis para determinar si existen diferencias significativas en alguna de las tareas entre los dos grupos de usuarios. La Tabla 8 muestra los resultados de los tests de Mann-Whitney ejecutados sobre los datos, en los cuales no se observan diferencias significativas entre programadores y no programadores para ninguna tarea, de modo que todos los participantes parecen haber trabajado de manera similar con estos controles.

Tarea	Mann-Whitney U	Z	p-value
Tarea 1	27,5	-0,474	0,636
Tarea 2	21,5	-1,104	0,269
Tarea 3	25,0	-0,737	0,461
Tarea 4	29,5	-0,265	0,791
Tarea 5	27,5	-0,475	0,635
Tarea 6	32,0	0,000	1,000
Tarea 7	20,5	-1,209	0,227
Tarea 8	31,0	-0,105	0,916

Tabla 8. Análisis estadístico del número de movimientos de colecciones para cada una de las tareas.

El último aspecto estudiado en esta dimensión está relacionado con la distribución de las colecciones dentro del espacio de trabajo durante la edición. Esta métrica debería ofrecer indicaciones visuales sobre cuál es el lugar más adecuado dentro del área de edición para situar un determinado tipo de colección, en función de las posiciones preferidas por los usuarios que han sido detectadas tras el análisis. Una redistribución de las colecciones en futuras versiones del editor permitiría que el proceso de edición fuese más cómodo, y reduciría el número de movimientos tanto para nodos como para colecciones, ya que las posiciones de los controles serían más cercanas a las necesidades de los usuarios desde un principio, evitando que el usuario tenga que desplazar los elementos para comprender mejor el proceso.

La vista inicial para definir una regla de comportamiento fuerza a los usuarios a situar los tangibles asociados a las colecciones del origen, evento y destinatario en puntos fijos de la pantalla, indicados con códigos de colores, para poder iniciar la exploración de las mismas. Estas posiciones iniciales para las colecciones del origen, evento y destinatario se muestran en la Figura 33, Figura 34 y Figura 35, respectivamente, como áreas circulares sombreadas. Desde esta posición inicial, los usuarios pueden desplazar los tangibles, y por tanto, las colecciones, hacia otros lugares de la pantalla. Se ha llevado a cabo un registro de todos los desplazamientos realizados con las colecciones, de modo que pudiese extraerse la posición final de cada desplazamiento, que es desde la que el usuario presumiblemente realiza la exploración de la colección. De este modo, la Figura 33, Figura 34 y Figura 35 muestran las posiciones finales para las tres colecciones nombradas diferenciando entre ambos grupos de usuarios. Puede observarse que, en su mayoría, los usuarios de ambos grupos se sienten cómodos con la posición inicial ofrecida por la herramienta para las tres colecciones, ya que los puntos de las posiciones finales de las mismas tienden a concentrarse en torno al área redondeada de la exploración inicial. Cabe destacar que las colecciones asociadas al origen y al evento, aunque se desplacen de posición, nunca son situadas en la mitad derecha de la pantalla. Para la colección de la población destino sí que se ha producido en contadas ocasiones un desplazamiento de la misma a la mitad izquierda de la pantalla, o incluso a la mitad inferior de la superficie, pero en su mayoría los usuarios han seguido prefiriendo la posición inicial sugerida para esta colección. Existen otras dos colecciones para las que no se sugiere ningún punto

inicial de posicionamiento, la colección de operadores y la colección de elementos ajenos. Llegados al punto de tener que utilizar estas dos colecciones se asume que los usuarios ya conocen el método de interacción con los tangibles, de modo que tienen total libertad para ubicar estos dos nuevos controles donde deseen. En la Figura 36 y Figura 37 se muestran las posiciones finales para las colecciones de operadores y elementos ajenos, respectivamente. Puede observarse que para la colección de elementos ajenos no ha sido posible detectar ninguna posición preferente, ya que en cada momento los usuarios la han ubicado donde quedaba espacio disponible en el área de edición, abarcando así toda la superficie. Pero en cuanto a la colección de operadores, resulta destacable que prácticamente la totalidad de manipulaciones hayan tenido lugar en la mitad inferior de la pantalla, y con mayor frecuencia en la esquina inferior derecha. La causa de esta predisposición por situar dicha colección cercana a la mano dominante de los usuarios puede encontrarse en estudios relacionados con la territorialidad de controles en interfaces basadas en superficies interactivas *(Ryall et al., 2004)(Scott et al., 2004)(Tse et al., 2004)*. Estos estudios indican que los usuarios tienden a ubicar cerca de ellos aquellos controles que usan con más frecuencia en un determinado momento. Si se tiene en cuenta que el uso de la colección de operadores es una interacción repetitiva a lo largo de todo el proceso de edición, y que todos los usuarios excepto uno indicaron ser diestros (el usuario restante era ambidiestro), se refuerzan los resultados obtenidos observando la Figura 37.

Como resumen de esta dimensión analizada, no se considera excesiva la reubicación de nodos por el área de edición, si bien deberían tomarse medidas para mejorar las ubicaciones automáticas de los nodos extraídos en futuras versiones del editor. Las posiciones de las colecciones del origen, evento y destinatario no parecen resultar incómodas para los usuarios. Para los controles de uso frecuente, la posición preferente sería la esquina inferior derecha de la pantalla.

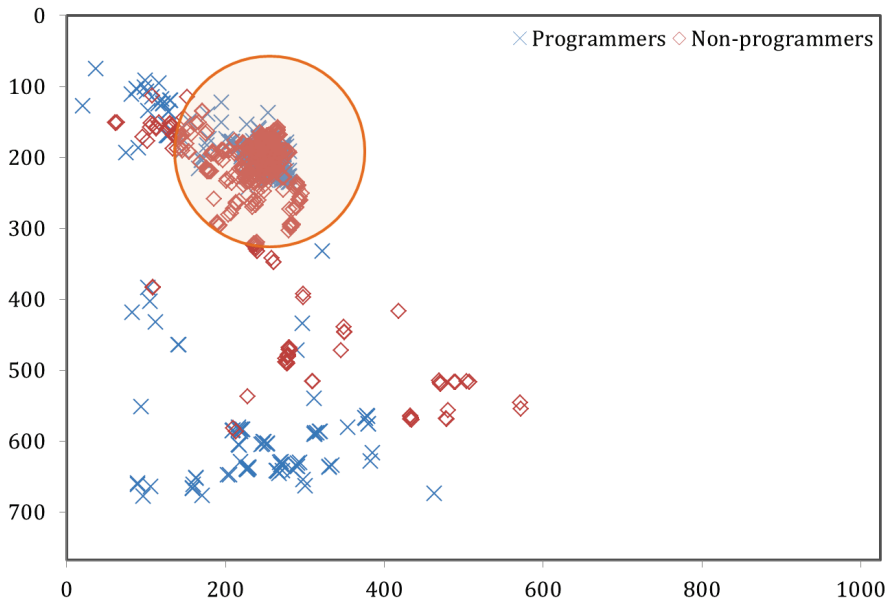


Figura 33. Posiciones finales para la colección de la población origen.

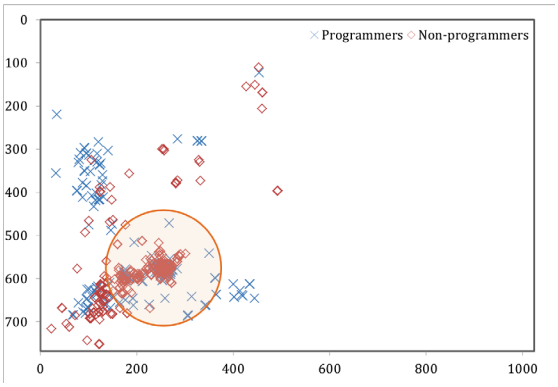


Figura 34. Posiciones finales para la colección de eventos.

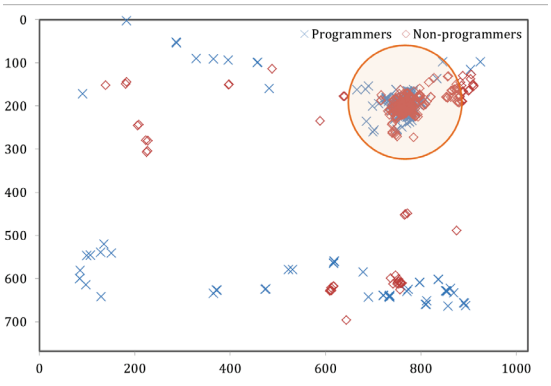


Figura 35. Posiciones finales para la colección de la población destino.

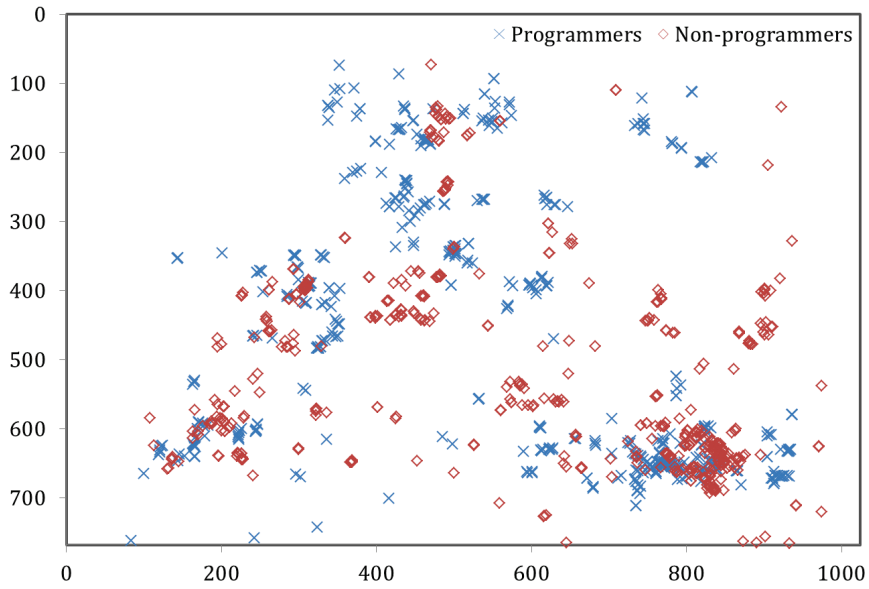


Figura 36. Posiciones finales para la colección de elementos ajenos.

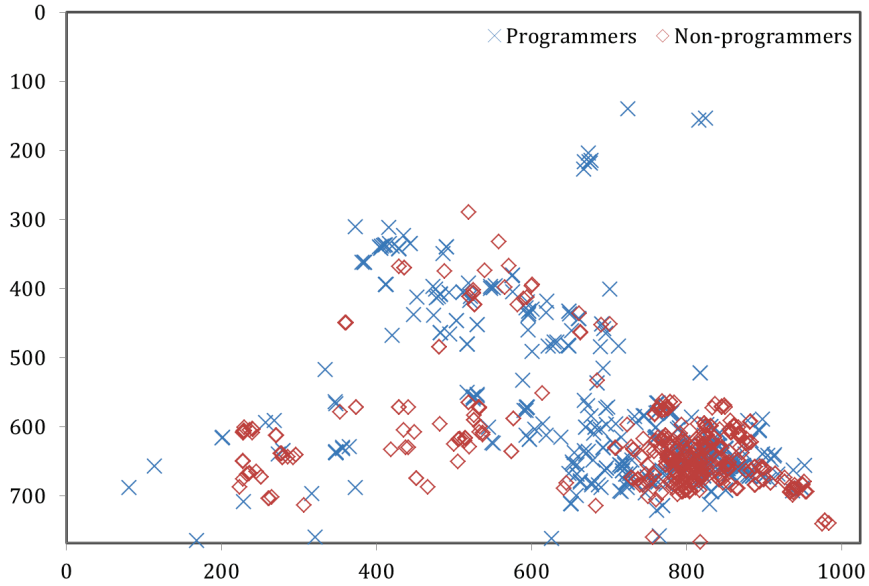


Figura 37. Posiciones finales para la colección de operadores.

5.2.7. Cuestionarios de usabilidad de la herramienta

Tras completar las tareas de edición con la herramienta, se solicitó a los participantes rellenar un cuestionario acerca de su experiencia y opinión personal con el editor. Las cuestiones versaban sobre distintos aspectos, entre ellos, la facilidad de uso, novedad de la herramienta propuesta, comprensión y adecuación de los mecanismos de interacción empleados, así como de los controles gráficos utilizados, entre otros. El objetivo principal del cuestionario era determinar si el editor había sido considerado como una herramienta efectiva para la especificación de comportamiento según los conocimientos previos de programación de cada usuario. El cuestionario comprendía un total de 15 preguntas a responder empleando una escala Likert de 5 puntos, junto a 6 preguntas de respuesta abierta. En el Anexo D se muestran los cuestionarios completos que fueron entregados a los participantes del experimento. A continuación se detallan los resultados de dichos cuestionarios, tanto de las preguntas de respuesta cerrada como de las sugerencias y comentarios de los usuarios en las preguntas de respuesta abierta.

El cuestionario comenzaba con un conjunto de 15 preguntas de respuesta cerrada, a responder por los usuarios utilizando una escala de puntuación Likert de 5 puntos, siendo el valor 1 correspondiente a la respuesta "*Totalmente en desacuerdo*", el 5 indicando que se está "*Totalmente de acuerdo*" con la cuestión, el valor 3 para indicar "*Ni de acuerdo ni en desacuerdo*", y las respuestas intermedias pertinentes para los valores 2 y 4. Las preguntas formuladas se muestran en la Tabla 9. En la Figura 38 se muestran las medianas de la puntuación obtenida en cada una de las preguntas evaluadas para cada grupo de participantes. En general, todos los aspectos evaluados fueron puntuados positivamente.

En primer lugar se pretendía evaluar si los mecanismos provistos para determinar los principales elementos de la regla eran fáciles de comprender. La selección de la población origen y destino, así como del evento de la regla (P1), han sido consideradas interacciones sencillas por ambos grupos de usuarios. Por otra parte, se ha evaluado cuán fácil resultaba para los usuarios añadir elementos a la expresión del proceso de datos a editar. En cuanto a la extracción de operadores al proceso de datos (P2), pese a los problemas de interacción detectados en la colección asociada, ambos grupos de usuarios han indicado que resultaba sencillo. Por su parte, la presencia de elementos ajenos en el proceso de datos era considerado uno de los factores influyentes en la dificultad percibida de una regla, y sin embargo los usuarios indicaron no haber encontrado mayor problema a la hora de añadir elementos ajenos al proceso (P3). La extracción de propiedades del origen y el destinatario (P4) también ha sido considerada por programadores y no programadores como una interacción sencilla. Cuando se pregunta a los usuarios por la adecuación del control de constantes numéricas para establecer el valor de las mismas (P5), las respuestas indican que este control no resulta tan sencillo de utilizar como se había diseñado.

Id	Pregunta: Considero que…
P1	Es fácil seleccionar el origen/suceso/destinatario de la regla.
P2	Es fácil añadir operadores nuevos al proceso de datos.
P3	Es fácil añadir elementos ajenos al proceso de datos.
P4	Es fácil añadir propiedades del origen y el destinatario al proceso de datos.
P5	Es sencillo establecer el valor de una constante numérica con el control en forma de regla utilizado.
P6	Conectar dos elementos con un flujo de datos es sencillo e intuitivo.
P7	Crear flujos de datos empleando los dedos resulta una tarea cómoda incluso teniendo que realizar numerosas conexiones.
P8	Es fácil eliminar elementos del proceso de datos.
P9	Resulta útil disponer de tangibles para explorar y posicionar las poblaciones del origen y el destinatario, así como el suceso.
P10	Preferiría realizar la tarea de edición en colaboración con otro usuario.
P11	El editor sería útil si tuviese una mesa interactiva en el contexto de una casa inteligente.
P12	El editor de reglas de comportamiento para superficies interactivas es una herramienta novedosa.
P13	El editor de reglas es fácil de utilizar.
P14	La gente podría aprender a utilizar el editor fácilmente.
P15	No necesitaría importante ayuda técnica para manejar el editor.

Tabla 9. Preguntas del cuestionario sobre la usabilidad de la herramienta.

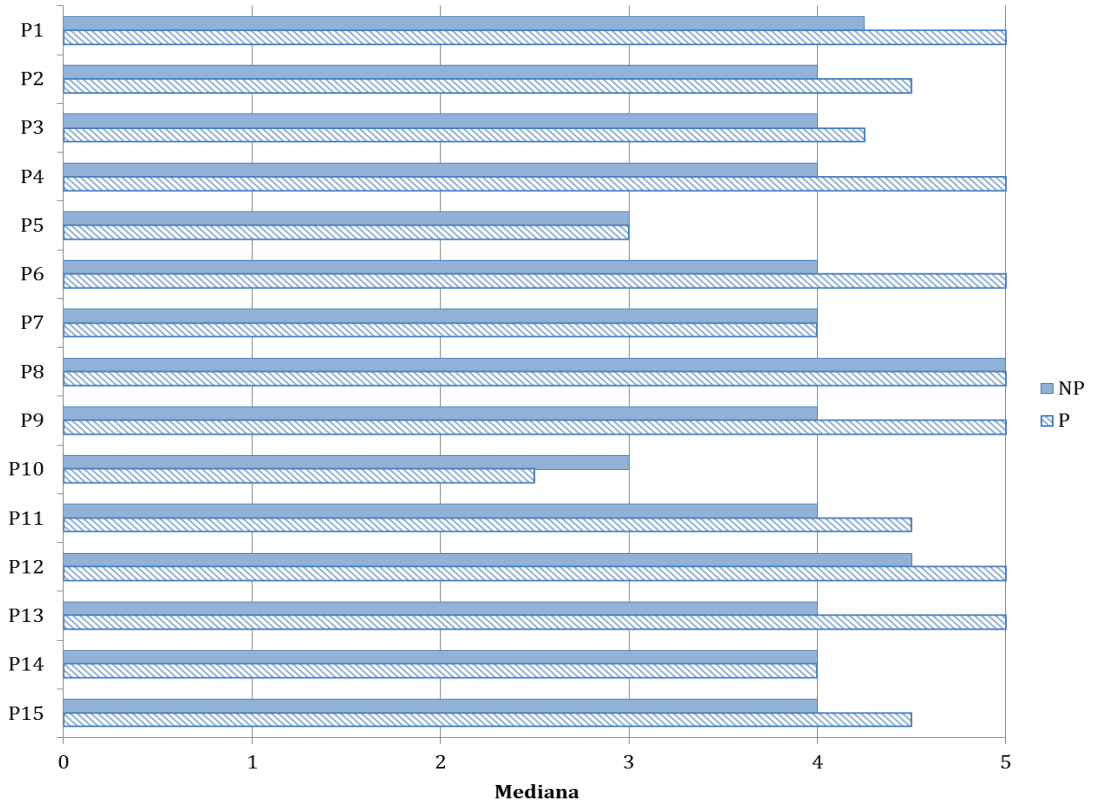


Figura 38. Mediana de las respuestas de cada grupo de usuarios ante el cuestionario de usabilidad de la herramienta.

La conexión de dos elementos del proceso de datos creando los flujos con los dedos (P6) ha resultado ser un mecanismo sencillo e intuitivo para todos los participantes. Más aún, los usuarios no han considerado como molesto tener que realizar las conexiones utilizando este mecanismo de interacción cuando el número de conexiones a crear es elevado (P7). Pero sin duda, la acción más fácil del proceso de edición ha resultado ser el borrado, ya que ambos grupos de usuarios valoran muy positivamente la sencillez de esta interacción (P8).

Un aspecto interesante a evaluar ha sido la percepción de los usuarios sobre los tangibles para la exploración y ubicación libre de las colecciones, en concreto de las asociadas al evento, origen y destinatario (P9). Todos los usuarios han considerado útil el uso de dichos tangibles con los propósitos especificados. Pero aunque los resultados han sido satisfactorios, la interacción con estas colecciones no debería verse ligada a la manipulación mediante tangibles, ya que quizás este mecanismo de interacción con las colecciones no esté disponible en otras plataformas sobre las que pueden desarrollarse futuras versiones del editor. En este caso, para seguir permitiendo libertad a los usuarios a la hora de posicionar las colecciones alrededor de la pantalla, la interacción debería realizarse completamente con los dedos.

Dada la capacidad de colaboración que ofrecen las superficies interactivas como la empleada en el experimento, y puesto que la interfaz de la herramienta se encuentra preparada para llevar a cabo la edición de reglas de manera cooperativa, se preguntó a los usuarios su opinión acerca de llevar a cabo la tarea de edición junto a otro usuario (P10). En el resto de preguntas del cuestionario, los programadores se mostraron más positivos que los no programadores, a excepción de esta cuestión. Los programadores han mostrado menor interés por la edición colaborativa que los no programadores, indicando una leve preferencia por realizar esta tarea solos. Por otra parte, los no programadores no dieron muestras de preferir ninguna de las dos opciones.

Se ha preguntado a los participantes por la utilidad percibida del editor en el supuesto de disponer de una casa inteligente con las capacidades necesarias para la definición de comportamiento, así como de una superficie interactiva sobre la que llevar a cabo la edición (P11). Las respuestas ofrecidas han mostrado que la herramienta sería considerada como efectiva en este supuesto. Por otro lado, la novedad de la herramienta de edición ha sido también altamente puntuada (P12) por ambos grupos de usuarios. En cuanto a la facilidad de uso percibida (P13), de nuevo tanto programadores como no programadores están de acuerdo en afirmar que el editor de reglas de comportamiento ha resultado fácil de manejar para las tareas propuestas.

Las dos últimas preguntas versaban sobre el proceso de aprendizaje necesario para manejar la herramienta. Los participantes consideran que es sencillo aprender a utilizar el editor (P14), y que para ello no sería necesario disponer de importante ayuda técnica para conseguir definir reglas de comportamiento como las realizadas (P15).

Id	Pregunta
PA1	¿Qué partes del proceso de edición te han resultado más claras/fáciles?
PA2	¿Qué partes del proceso de edición te han resultado más complejas/difíciles?
PA3	¿Destacarías algo que haya sido especialmente agotador al manejar la herramienta?
PA4	¿Cómo crees que podría mejorarse el proceso de conexión de flujos de datos?
PA5	Si las colecciones asociadas al origen, evento y destinatario tuviesen que estar en una posición fija, ¿dónde las ubicarías?
PA6	Comentarios/Sugerencias de mejoras

Tabla 10. Preguntas de respuesta abierta.

Por otra parte, los usuarios debían responder las 6 preguntas de respuesta abierta que se muestran en la Tabla 10. Estas preguntas permiten que los usuarios ofrezcan su opinión personal y comentarios sobre su experiencia con la herramienta de edición utilizada. En primer lugar, se solicitó a los participantes que indicasen qué partes del proceso de edición les habían resultado más sencillas (PA1), complejas (PA2) y agotadoras (PA3). El aspecto considerado como más sencillo del proceso de edición parece ser la selección de las poblaciones origen y destino, así como el evento, ya que 8 usuarios lo indicaron así en sus respuestas, mientras que en segundo lugar queda la conexión de elementos empleando flujos de datos, indicado por 4 usuarios. El aspecto más complejo del editor ha resultado ser el establecimiento de constantes numéricas, tal y como reportan 7 usuarios, seguido por la exploración de la colección de operadores para extraer el operador necesario, indicado en 3 ocasiones. En cuanto a aspectos agotadores de la sesión con la herramienta, estos han tenido que ver más bien con la infraestructura de la mesa interactiva empleada que con aspectos de usabilidad de la herramienta, ya que 5 usuarios indicaron que la baja altura de la mesa les obligaba a tomar una postura incómoda.

Cuando se les preguntó a los usuarios cómo podría llevarse a cabo la conexión de flujos de datos de manera más óptima (PA4), 8 de los participantes propusieron que las uniones se llevasen a cabo tan sólo tocando los dos nodos a conectar, bien con el dedo o bien mediante tangibles específicos, y no realizando el dibujado de la línea de conexión a lo largo de la superficie. Otra respuesta, ofrecida por 3 usuarios, indicaba que la bidireccionalidad a la hora de establecer las conexiones sería deseable. Es decir, que los nodos pudiesen conectarse creando flujos de datos tanto en el sentido natural en el que deberían fluir los datos, desde los nodos productores (propiedades, constantes numéricas, etc.) hacia los nodos consumidores (entradas de operadores, nodo destino), como en el sentido contrario. Otra de las respuestas proponía como nuevo mecanismo de creación de flujos de datos la conexión por proximidad entre dos nodos.

Se solicitó a los usuarios que indicasen, en caso de tener que seleccionar una ubicación fija para las colecciones del origen, destinatario y evento, la posición más adecuada en cada caso según su criterio (PA5). Las respuestas a esta cuestión han

sido de lo más variadas. Si bien todos los participantes indican que las colecciones fijas deberían ubicarse en uno o varios de los cuatro laterales de la pantalla, cada uno de ellos proponía una disposición diferente: algunos indicaban que preferían las tres colecciones en el mismo lateral, mientras que otros afirmaban que sería más conveniente que cada colección estuviese en un lateral distinto. Y para mayor variedad todavía, algunos de ellos optaban por hacerlas visibles constantemente mientras que otros preferían colecciones desplegables.

Finalmente, la última cuestión permitía a los usuarios ofrecer cualquier comentario o sugerencia que considerasen de interés a tener en cuenta para el experimento y para posibles mejoras en futuras versiones del editor (PA6). A raíz de estos comentarios han surgido diversas ideas interesantes, algunas de las cuales ya se habían considerado de manera previa al experimento, mientras que otras aportaciones han resultado abrir interesantes vías de discusión. Por ejemplo, algunas sugerencias de los usuarios han permitido tomar ideas para el rediseño del control de inserción de constantes numéricas, o también para solucionar los problemas detectados con la colección de operadores. Otro tipo de sugerencias versaban sobre características no disponibles en la interfaz actual, que podría ser interesante incluir si se detecta su necesidad en próximos experimentos, como por ejemplo, disponer de alguna opción para volver al estado inmediatamente anterior de una colección, o poder modificar un elemento que ya haya sido añadido a la expresión directamente desde el área de edición.

5.2.8. Discusión de resultados

Tras el experimento llevado a cabo con la herramienta de edición de reglas de comportamiento se han estudiado distintas métricas relacionadas con el rendimiento del proceso de edición, la corrección de las reglas y la disposición de los elementos en la interfaz. Además, se han revisado las respuestas de los usuarios a los cuestionarios posteriores al experimento. Tras el análisis estadístico y de los cuestionarios se presenta a continuación un resumen y discusión de los resultados derivados del estudio.

Los resultados obtenidos tras el experimento han permitido determinar que tanto usuarios programadores como no programadores son capaces de manejar los conceptos subyacentes en los que se basa la herramienta de edición sin dificultades destacables. Además, se ha demostrado que ambos grupos de usuarios también son capaces de utilizar los mecanismos de interacción ofrecidos por el editor para llevar a cabo la definición de reglas de comportamiento complejas.

En primer lugar, aunque los programadores completaron todos los ejercicios en menos tiempo que los no programadores, tal y como cabía esperar, el tiempo medio empleado por cada grupo de usuarios únicamente difiere cuando se trata de editar operaciones complejas. Además, todos los usuarios fueron capaces de completar las tareas en un intervalo de tiempo razonable, teniendo en cuenta que se trataba de la primera vez que se enfrentaban a un problema de este tipo y la primera vez que manejaban una herramienta diseñada para superficies interactivas. Con todo ello en cuenta, es remarcable que ambos grupos de usuarios

considerasen, según reflejan los cuestionarios, que el editor resultaba fácil de utilizar y fácil de aprender sin importante ayuda técnica.

En segundo lugar, para las dificultades en el proceso de edición que se han detectado, ambos grupos de usuarios se comportan igual. Por ejemplo, en cuanto a exploraciones indeseadas o al borrado de elementos del proceso de datos, ambos grupos de usuarios han experimentado problemas con el mismo tipo de colección o de control, de modo que resulta fácil identificar los elementos de la interfaz que requieren de un rediseño o replanteamiento para evitar estos errores.

En tercer lugar, a pesar de haberse detectado errores en las reglas editadas, ha sido fácil realizar una clasificación de los mismos, ya que los diversos tipos de error producidos estaban acotados. Únicamente dos usuarios cometieron un elevado número de errores, las causas de los cuales han sido analizadas y justificadas. Aun con ello, los resultados demuestran que no existen diferencias entre programadores y no programadores en cuanto a la cantidad de errores cometidos.

Por otro lado, se ha observado que ambos grupos de participantes se comportan de manera similar cuando se trata de manipular elementos en la interfaz. Todos los usuarios tienden a mover y redistribuir nodos en el área de edición repetidas veces, manteniendo una disposición de los elementos estructurada dentro del mismo ejercicio. Esto demuestra que son capaces de manejarse correctamente con las técnicas de interacción directa con los dedos que propone la interfaz, ya que son capaces de reubicar los elementos a su gusto. Una característica altamente deseable en el proceso de edición visual ha sido la posibilidad de manipular libremente las colecciones empleando tangibles. Los usuarios han indicado que se trata de un mecanismo muy útil tanto para la exploración como para el desplazamiento de colecciones a fin de manejar mejor el espacio disponible. Además, es posible detectar patrones de posicionamiento, tanto mediante el estudio de las sesiones grabadas en vídeo como observando las posiciones finales donde los usuarios han ubicado las colecciones. Puede observarse que la mayoría de usuarios han escogido la misma posición para manejar las colecciones de poblaciones origen y destino, evento y operadores.

Las técnicas de interacción para editar flujos de datos entre dos elementos han sido consideradas como fáciles e intuitivas, pero los cuestionarios de respuesta abierta han ofrecido algunas sugerencias de mejora a tener en cuenta. Resultaría interesante disponer de bidireccionalidad al establecer las conexiones entre nodos, además de poder conectar elementos tocando únicamente los nodos involucrados mediante tangibles o con el dedo.

Sin embargo, se han detectado algunos problemas de usabilidad, y se han analizado distintas soluciones posibles a los mismos. En el editor empleado para el experimento, la representación de los elementos del ecosistema empleando sus nombres textuales dentro de las colecciones ha resultado inadecuada, tal y como demuestran los cuestionarios de opinión de los usuarios, así como los datos acerca de errores cometidos por los usuarios durante la edición. Diversos errores en la definición de las reglas han sido fruto de la confusión de elementos cuyos nombres son similares, como por ejemplo la selección de propiedades o poblaciones cuyos

nombres son semejantes, e incluso la conexión incorrecta de flujos de datos. Por este motivo, se hace patente la necesidad de una representación más visual para los elementos del ecosistema. Los tipos de las entidades dentro de una casa inteligente son fáciles de identificar empleando una imagen en miniatura de los mismos, tales como imágenes de una televisión o una radio. Sin embargo, no se ha encontrado solución todavía para diferenciar visualmente entre entidades de un mismo tipo, p.ej., *RadioOficina* y *RadioCocina*. Las soluciones propuestas varían desde cambiar la orientación de los textos hasta combinar representaciones visuales y textuales. La colección de operadores ha sido detectada como otro de los focos de errores de usabilidad de la herramienta. Puesto que los operadores se clasifican por categorías, y se ha optado por representar dichas categorías de manera textual en la colección, en ocasiones a los usuarios no les resulta obvio distinguir dentro de qué categoría se encuentra el operador deseado. En consecuencia, se han producido excesivas exploraciones indeseadas con esta colección. Una solución eficiente consistiría en asociar como representación visual de una categoría de operadores, la miniatura de aquellos contenidos en la misma. Además, la representación visual de los operadores "*mayor que*" y "*menor que*" debe ser cambiada por otra más intuitiva, ya que dichos operadores han sido la causa de diversos errores en el proceso de edición, y hasta cinco usuarios reportaron en los cuestionarios que no era posible diferenciarlos claramente. La posibilidad de colaboración entre distintos usuarios para editar una regla de comportamiento ha llevado a utilizar controles 360 grados y textos reflejados, lo cual induce a confusión tanto con la colección de operadores como con la representación de los elementos del ecosistema. Es por ello que debería reconsiderarse si la herramienta va a ser mayormente utilizada por un único usuario, evitando entonces representaciones orientadas a facilitar la edición en 360 grados, y orientando entonces los elementos hacia la posición del usuario.

Otro de los problemas de usabilidad detectados tiene que ver con el control de inserción de constantes numéricas. Esto es observable tanto en los cuestionarios, donde los usuarios reportaron su insatisfacción con dicho control, como observando el número de constantes numéricas que han tenido que ser borradas durante la edición de las tareas. Como resultado, este control debería ser rediseñado. Hay diversas soluciones propuestas por los usuarios, algunas de las cuales ya se habían considerado e implementado, como por ejemplo la mostrada en la Figura 39. Con este control se utilizaría la rotación natural de los tangibles para establecer el valor de una constante numérica, en lugar del slider del que se dispone actualmente. Otra de las soluciones consiste en emplear los símbolos "+" y "-", como se muestra en la Figura 40, para incrementar o disminuir el valor a establecer. La última solución considerada sería emplear un control con apariencia de calculadora para introducir los números, como en la Figura 41.

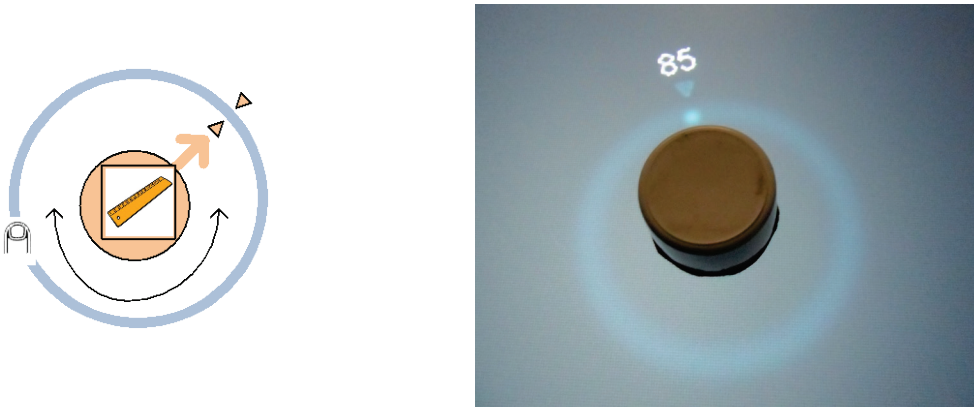


Figura 39. Control de insercion de constantes numéricas mediante rotación del tangible.

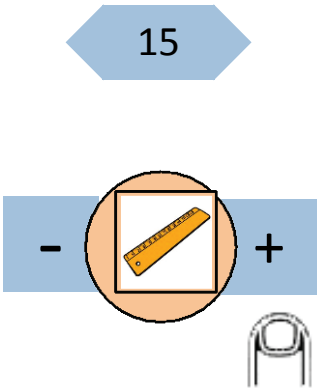


Figura 40. Control de inserción de constantes numéricas mediante incrementos y decrementos.

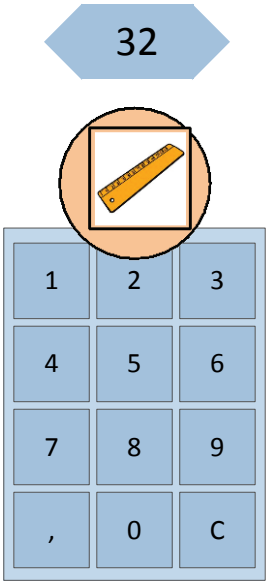


Figura 41. Control de inserción de constantes numéricas a modo de calculadora.

6. Conclusiones y trabajo futuro

6.1. Conclusiones

La personalización del comportamiento en entornos inteligentes requiere de lenguajes tanto lo suficientemente expresivos para permitir el control total del sistema, como comprensibles, para que los usuarios finales sean capaces de llevar a cabo esta tarea sin problemas, independientemente de su experiencia previa en el campo de la computación. Además, los mecanismos que permitan a los usuarios llevar a cabo la definición de comportamiento deben resultar intuitivos y adecuados, para ayudar a abstraer la complejidad del lenguaje de definición de comportamiento empleado.

En este trabajo se ha presentado una herramienta de edición de comportamiento basada en superficies interactivas, cuyo objetivo es permitir la definición de reglas reactivas para entornos inteligentes por parte de los usuarios finales del sistema. La especificación de comportamiento en esta herramienta se lleva a cabo empleando un modelo de reglas genérico enriquecido mediante expresiones de flujos de datos. Todo ello permite crear reglas altamente expresivas empleando una representación comprensible por usuarios sin nociones previas de programación. Se ha llevado a cabo un experimento para evaluar la usabilidad y adecuación del editor de reglas desarrollado para soportar la definición de comportamiento en entornos inteligentes por parte de usuarios con distintos niveles de conocimientos sobre programación. Los resultados obtenidos han permitido demostrar que, en términos generales, usuarios sin nociones previas de programación han sido capaces de especificar reglas de comportamiento satisfactoriamente empleando el lenguaje visual propuesto y la herramienta presentada, obteniéndose diferencias mínimas en relación con la ejecución de los usuarios programadores.

No obstante, el experimento ha permitido revelar algunos problemas de usabilidad de la herramienta, tales como el inadecuado uso de representaciones textuales para los elementos, la confusa clasificación de operadores en categorías y la poca efectividad del control para la inserción de constantes numéricas. En cambio, otros mecanismos de interacción propuestos han sido evaluados como sencillos y útiles para su propósito, como por ejemplo la exploración de colecciones empleando tangibles o la creación de flujos de datos empleando los dedos.

Por tanto, el editor de reglas de comportamiento desarrollado se muestra como una herramienta prometedora para su futuro uso por usuarios finales en ambientes inteligentes, ofreciendo grandes posibilidades de integración con diversos tipos de espacios inteligentes gracias a la generalidad del modelo de reglas en el que se basa.

6.2. Trabajo futuro

En cuanto a tareas pendientes con respecto al experimento conducido queda el análisis detallado de los vídeos de las sesiones de experimentación para tratar de detectar patrones de comportamiento. Por otro lado, los ejercicios en papel que los participantes realizaron previos al uso de la herramienta están siendo analizados. Puesto que la especificación de las reglas a editar con la herramienta ya se entregaba a los usuarios en papel, la evaluación de la herramienta ha permitido centrarse en la detección de problemas de usabilidad con respecto a la misma, y en la comprensibilidad del modelo de reglas que permiten los elementos de la interfaz gráfica desarrollados. Estudios previos de este grupo de investigación también permitieron determinar que las expresiones visuales basadas en flujos de datos eran fácilmente comprensibles por usuarios no programadores. Pero los ejercicios en papel resultantes de la introducción al modelo de reglas completo permitirán detectar si tanto programadores como no programadores son capaces de entender y especificar desde cero sus propias reglas de comportamiento con la notación empleada, y qué aspectos del modelo son más costosos de comprender.

Como trabajo pendiente para futuras versiones del editor quedaría resolver los problemas de usabilidad detectados, rediseñando los controles de la interfaz que han sido la causa de error durante la edición de reglas con la herramienta.

Adicionalmente, algunos mecanismos de interacción no reportados como problemáticos están siendo también reconsiderados, como por ejemplo la creación de flujos de datos empleando los dedos para crear las conexiones. Esta forma de definir un flujo de datos resulta intuitiva pero introduce tiempos de manipulación más largos, sobre todo cuando hay numerosas conexiones que realizar. Además, la manipulación de elementos de la superficie directamente con los dedos introduce problemas de precisión debidos a la oclusión de los elementos de la pantalla con la misma mano o brazo del usuario *(Vogel & Casiez, 2012)*. Con el mecanismo actual de creación de flujos de datos el usuario puede fallar al tratar de realizar la conexión, ya que al dibujar la línea de conexión la mano o dedo del usuario tapan el nodo destino del flujo de datos. Si esto se produce, puede que el usuario levante el contacto pensando que su dedo está sobre el nodo deseado, cuando en realidad no es así. En consecuencia, se pretende reducir el tiempo empleado en la creación de estos flujos de datos introduciendo nuevas y mejoradas técnicas de conexión, así como mejorar la precisión de las mismas. En el contexto del grupo de investigación ISSI se han desarrollado y evaluado un conjunto de técnicas genéricas para la conexión de nodos *(Zorrilla, 2013)* que podrían incorporarse en futuras versiones del editor. Entre estas técnicas se incluye la conexión punto a punto solicitada por los usuarios en los cuestionaros de usabilidad, en la cual un flujo de datos es creado tocando con el dedo los dos elementos involucrados. Otras técnicas implementadas permiten la conexión basada en la proximidad, en un caso desplazando el nodo origen hacia el nodo destino de la conexión, y en otro caso basándose en la dirección de la línea que el usuario dibuja para crear el flujo de datos, tomando como nodo destino el más próximo siguiendo dicha dirección. También se ha desarrollado una novedosa técnica que incorpora un visor de la superficie ocluida mientras se está llevando a cabo la conexión. Se desea realizar

experimentos con un mayor número de usuarios para probar el potencial y efectividad de todas las técnicas mencionadas, de modo que puedan incorporarse al editor las más adecuadas.

Como característica necesaria a incorporar en nuevas versiones del editor también se presenta la posibilidad de trabajar con acciones que incluyan más de un parámetro. En la versión actual de la herramienta, los usuarios pueden escoger si la operación afectará a una propiedad de la población destino, o si bien conllevará la ejecución de una acción. Pero puesto que únicamente hay una vista para el proceso de datos de la operación, se ha simplificado la interfaz permitiendo únicamente acciones con un parámetro. Resulta necesario encontrar algún método intuitivo para permitir ejecutar acciones que involucren dos o más parámetros, permitiendo en la interfaz alternar entre la edición de los procesos de datos de cada parámetro, sin alterar el resto de controles y sin sobrecargar la interfaz con excesivos conceptos.

Aunque la herramienta de edición haya sido útil para el propósito con el que se ha desarrollado, los usuarios no expertos de la misma podrían experimentar dificultades en sus primeros intentos de crear sus propias reglas de comportamiento. Incluso usuarios expertos podrían sentirse frustrados enfrentándose a la edición de reglas altamente complejas y expresivas partiendo de una especificación en blanco. Por tanto, resulta interesante investigar qué tipo de asistencia podría ofrecerse al usuario durante el proceso de personalización de su entorno para evitar este tipo de situaciones, que en ocasiones conllevan el rechazo de la tecnología por la falta de comprensión de la misma. Se ha comenzado a plantear el diseño de un sistema de gestión inteligente de reglas de comportamiento, destinado a facilitar la configuración efectiva de entornos inteligentes por parte de usuarios no expertos mediante la asistencia guiada en el proceso de edición. Esta edición asistida de reglas debería proveer mecanismos inteligentes para anticiparse a las interacciones de los usuarios, ofreciéndoles sugerencias o correcciones basadas en el análisis de reglas previamente definidas en un repositorio global, empleando heurísticas para la detección de patrones en las mismas, etc. Se trata de un proyecto ambicioso, ya que se deberían explorar distintas dimensiones, desde qué tipo de patrones o heurísticas son los más adecuados según las características del usuario o del entorno en el que se vaya a emplear el sistema, hasta cómo deben mostrarse las sugerencias o auto-compleciones en la interfaz gráfica para evitar la creación de un sistema intrusivo.

Otro trabajo futuro interesante en relación con la evolución del entorno en una situación real tiene que ver con el manejo de reglas en conflicto. Los entornos inteligentes estarán poblados de diversos usuarios, cada uno de los cuales tendrá preferencias específicas y distintas. Si cada uno de estos usuarios define sus propias reglas de comportamiento en el mismo entorno, probablemente aparezcan reglas contradictorias o en conflicto entre sí. Hay trabajos relacionados en este ámbito que tratan de definir sistemas multiagentes para manejar las preferencias de distintos usuarios conviviendo en un mismo entorno *(García-Herranz, Alamán, et al., 2010)*. Incluso un único usuario puede especificar comportamiento contradictorio entre sus propias reglas sin darse cuenta. Por este motivo, es necesario proveer mecanismos para la detección automática de reglas en conflicto.

Además, debería estudiarse la manera de ofrecer al usuario retroalimentación visual acerca de los problemas detectados y sus posibles soluciones, para asistir al usuario en la corrección de estos conflictos. Existen trabajos emergentes en cuanto a *inteligibilidad*, es decir, la capacidad de comprender cómo modelan los usuarios el comportamiento de su entorno y cómo representan la información con el fin de permitirles manejar este tipo de situaciones en las que el sistema no responde como esperan *(Dey, 2009)*.

Finalmente, otras vías prometedoras de trabajo serían la integración y evaluación del sistema en un entorno real, así como explorar nuevas plataformas sobre las que desplegar la herramienta, tales como tabletas u otros dispositivos táctiles y/o tangibles. También se plantea el refinamiento del modelo de reglas así como de la herramienta de edición asociada, adaptándolos a un DSL concreto del área de la Inteligencia Ambiental. Esto permitiría obtener un mayor grado de expresividad en las reglas, siendo capaces de integrar construcciones inherentemente necesarias en el ámbito AmI, tales como lógicas temporales o espaciales.

6.3. Publicaciones

El presente trabajo ha derivado en los siguientes artículos de investigación:

- Patricia Pons, Alejandro Catalá, Javier Jaén. *Customizing Smart Environments: A tabletop approach*. Journal of Ambient Intelligence and Smart Environments (JCR 2012, IF: 1.298, Computer Science – Information Systems T1 [43/132]), pendiente de aceptación.
- Alejandro Catalá, Patricia Pons, Javier Jaén, Jose Antonio Mocholí, Elena Navarro. *A Meta-Model for DataFlow-Based Rules in Smart Environments: Evaluating User Comprehension and Performance*. Science of Computer Programming Journal, vol. 78 (10), pp. 1930-1950, Octubre 2013 (JCR 2010, IF: 1.306, Computer Science – Software Engineering T1 [32/99]).
- Patricia Pons, Alejandro Catalá, Javier Jaén. *TanRule: A Rule Editor for Behavior Specification on Tabletops*. Extended Abstracts of the ACM Tangible, Embedded and Embodied Interaction (TEI 2013), Febrero 2013.

Además del trabajo anterior se han derivado diversos artículos, fruto de estudios previos sobre el modelo de reglas reactivas enriquecidas mediante flujos de datos, la validación del prototipo de la herramienta, el middleware para la gestión de reglas reactivas basadas en el modelo propuesto y su uso en contextos de juegos reactivos para superficies interactivas, así como otros estudios relacionados con el uso de interfaces tangibles de usuario:

- Alejandro Catalá, Javier Jaén, Patricia Pons, Fernando García. *Creativity and Entertainment: Experiences and Future Challenges.* I Simposio Español de Entretenimiento Digital (SEED ’13), Septiembre 2013.
- Alejandro Catalá, Fernando García, Patricia Pons, Javier Jaén, José Antonio Mocholí. *AGORAS: Towards collaborative game-based learning experiences*

on surfaces. Proceedings of the International Conference on Cognition and Exploratory Learning (CELDA 2012), pp. 147-154, IADIS, Octubre de 2012.

- Alejandro Catalá, Javier Jaén, Fernando García, Patricia Pons. *Ownership and Dominance in Tabletop Based Collaboration for Creative Tasks*. Actas de las Jornadas sobre aprendizaje colaborativo en entornos virtuales de la red. Red temática sobre Aprendizaje Colaborativo en Entornos Virtuales (RACEV), 2012.
- Patricia Pons, Alejandro Catalá, Javier Jaén, Jose Antonio Mocholí. *DaFRule: Un Modelo de Reglas Enriquecido mediante Flujos de Datos para la Definición Visual del Comportamiento Reactivo de Entidades Virtuales*. Actas de las Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2011), pp. 989-1002, 2011.
- Alejandro Catalá, Patricia Pons, Javier Jaén, Jose Antonio Mocholí. *Evaluating User Comprehension of DataFlows in Reactive Rules for Event-Driven AMI Environments*. Proceedings of the V International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'11), pp. 337-344, 2011.

7. Bibliografía

Becker, C., Handte, M., Schiele, G. & Rothermel, K., PCOM - a component system for pervasive computing. En *PerCom04 Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications*. IEEE, pp. 67–76, 2004.

Beckmann, C. & Dey, A.K., SiteView : Tangibly Programming Active Environments with Predictive Visualization. *Technical Report IRB-TR-03-019, Intel Research Berkeley*, 2003.

Bonino, D., Corno, F. & Russis, L. De, A user-friendly interface for rules composition in intelligent environments. *Ambient Intelligence Software and Applications*, 92, pp.213–217, 2011.

Catalá, A., Pons, P., et al., A meta-model for dataflow-based rules in smart environments: Evaluating user comprehension and performance. *Science of Computer Programming*, 78(10), pp.1930–1950, 2013.

Catalá, A., García, F., Pons, P., et al., AGORAS: Towards collaborative game-based learning experiences on surfaces. En *Proceedings of the International conference on cognition and Exploratory Learning CELDA 2012*. pp. 147–154, 2012.

Catalá, A., *AGORAS: Towards educational places for action, discussion and reflection to support creative learning on interactive surfaces*. Tesis doctoral, Universidad Politécnica de Valencia. 2012.

Catalá, A., García, F., Jaen, J. & Mocholi, J.A., TangiWheel: A Widget for Manipulating Collections on Tabletop Displays Supporting Hybrid Input Modality. *Journal of Computer Science and Technology*, 27(4), pp.811–829, 2012.

Catalá, A., Jaen, J., Pons, P. & García, F., Creativity and Entertainment : Experiences and Future Challenges. En *I Simposio Español de Entrenimiento Digital SEED 2013*. 2013.

Churchill, E.F., Putting the Person Back into Personalization. *Interactions*, 20(5), pp.12–15, 2013.

Cook, D.J., Augusto, J.C. & Jakkula, V.R., Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4), pp.277–298, 2009.

Dey, A.K. et al., a CAPpella: programming by demonstration of context-aware applications. En *Proceedings of the 2004 Conference on Human Factors in Computing Systems CHI 04*. ACM Press, pp. 33–40, 2004.

Dey, A.K., Modeling and intelligibility in ambient environments. *Journal of Ambient Intelligence and Smart Environments*, 1(1), pp.57–62, 2009.

Dey, A.K., Sohn, T., Streng, S. & Kodama, J., iCAP: Interactive prototyping of context-aware applications. En *Proceedings of Pervasive Computing*. Springer, pp. 254–271, 2006.

Fuentes, L., Jimnez, D. & Pinto, M., Development of Ambient Intelligence Applications using Components and Aspects. *Journal Of Universal Computer Science*, 12(3), pp.236–251, 2006.

Gámez, N. & Fuentes, L., FamiWare: a family of event-based middleware for ambient intelligence. *Personal and Ubiquitous Computing*, 15(4), pp.329–339, 2011.

García-Herranz, M., Alamán, X. & Haya, P., Easing the Smart Home: A rule-based language and multi-agent structure for end user development in Intelligent Environments. *Journal of Ambient Intelligence and Smart Environments*, 2(4), pp.437–438, 2010.

García-Herranz, M., Haya, P.A. & Alamán, X., Towards a Ubiquitous End-User Programming System for Smart Spaces. *Journal Of Universal Computer Science*, 16(12), pp.1633–1649, 2010.

Good, J., Howland, K. & Nicholson, K., Young people's descriptions of computational rules in role-playing games: An empirical study. En *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing VLHCC 2010*. IEEE Computer Society, pp. 67–74, 2010.

Holloway, S. & Julien, C., The case for end-user programming of ubiquitous computing environments. En *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*. ACM Press, pp. 167–172, 2010.

Ishii, H. & Ullmer, B., Tangible Bits : Towards Seamless Interfaces between People, Bits and Atoms. En *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. ACM Press, pp. 234–241, 1997.

Kelleher, C. & Pausch, R., Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2), pp.83–137, 2005.

López de Ipiña, D., An eca rule-matching service for simpler development of reactive applications. *Middleware*, 2(7), 2001.

Maternaghan, C. & Turner, K.J., A configurable telecare system. En *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments - PETRA '11*. New York, New York, USA: ACM Press, pp. 14:1–14:8, 2011.

Norman, D.A., *The invisible computer*, Cambridge, MA, USA: MIT Press. 1998.

Pane, J., Myers, B.A. & Ratanamahatana, C.A., Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54(2), pp.237–264, 2001.

Pons, P., *DafRule: Un modelo de reglas enriquecido mediante flujos de datos para la definición visual de comportamiento en entornos reactivos*. Proyecto Final de Carrera, Universidad Politécnica de Valencia. 2012.

Pons, P., TanRule : A Rule Editor for Behavior Specification on Tabletops. En *Extended Abstracts of the ACM Tangible, Embedded and Embodied Interaction TEI 2013*. pp. 1–8, 2013.

Pons, P., Catalá, A., Jaén, J. & Mocholí, J.A., DaFRule : Un Modelo de Reglas Enriquecido mediante Flujos de Datos para la Definición Visual del Comportamiento Reactivo de Entidades Virtuales. En *Actas de las Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2011*. pp. 989–1002, 2011.

Ryall, K., Forlines, C., Shen, C. & Morris, M.R., Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. En *Proceedings of the 2004 ACM conference on Computer supported cooperative work CSCW 04*. ACM Press, pp. 284–293, 2004.

Schmidt, A., Implicit human computer interaction through context Y. Nakhimovsky & J. Riegelsberger, eds. *Personal Technologies*, 4(2-3), pp.191–199, 2000.

Schöning, J. et al., Multi-Touch Surfaces: A Technical Guide. *Technical Report TUMI0833, Technical Reports of the Technical University of Munich*, 2008.

Scott, S.D., Sheelagh, M., Carpendale, T. & Inkpen, K.M., Territoriality in collaborative tabletop workspaces. En *Proceedings of the 2004 ACM conference on Computer supported cooperative work CSCW 04*. ACM Press, pp. 294–303, 2004.

Shadbolt, N., Ambient Intelligence. *IEEE Intelligent Systems*, 18(4), pp.2–3, 2003.

Teruel, M. a. et al., Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments. *Information and Software Technology*, 54(11), pp.1215–1228, 2012.

Tse, E., Histon, J., Scott, S.D. & Greenberg, S., Avoiding Interference: How People Use Spatial Separation and Partitioning in SDG Workspaces. En *Proceedings of the 2004 ACM conference on Computer supported cooperative work CSCW 04*. ACM Press, pp. 252–261, 2004.

Uribarren, A. et al., A Middleware Platform for Application Configuration, Adaptation and Interoperability. *2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pp.162–167, 2008.

Vogel, D. & Casiez, G., Hand occlusion on a multi-touch tabletop. En *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, New York, USA: ACM Press, pp. 2307–2316, 2012.

Weis, T., Handte, M., Knoll, M. & Becker, C., Customizable Pervasive Applications. En *Proceedings of the Fourth Annual IEEE International Conference on*

Pervasive Computing and Communications. IEEE Computer Society, pp. 239–244, 2006.

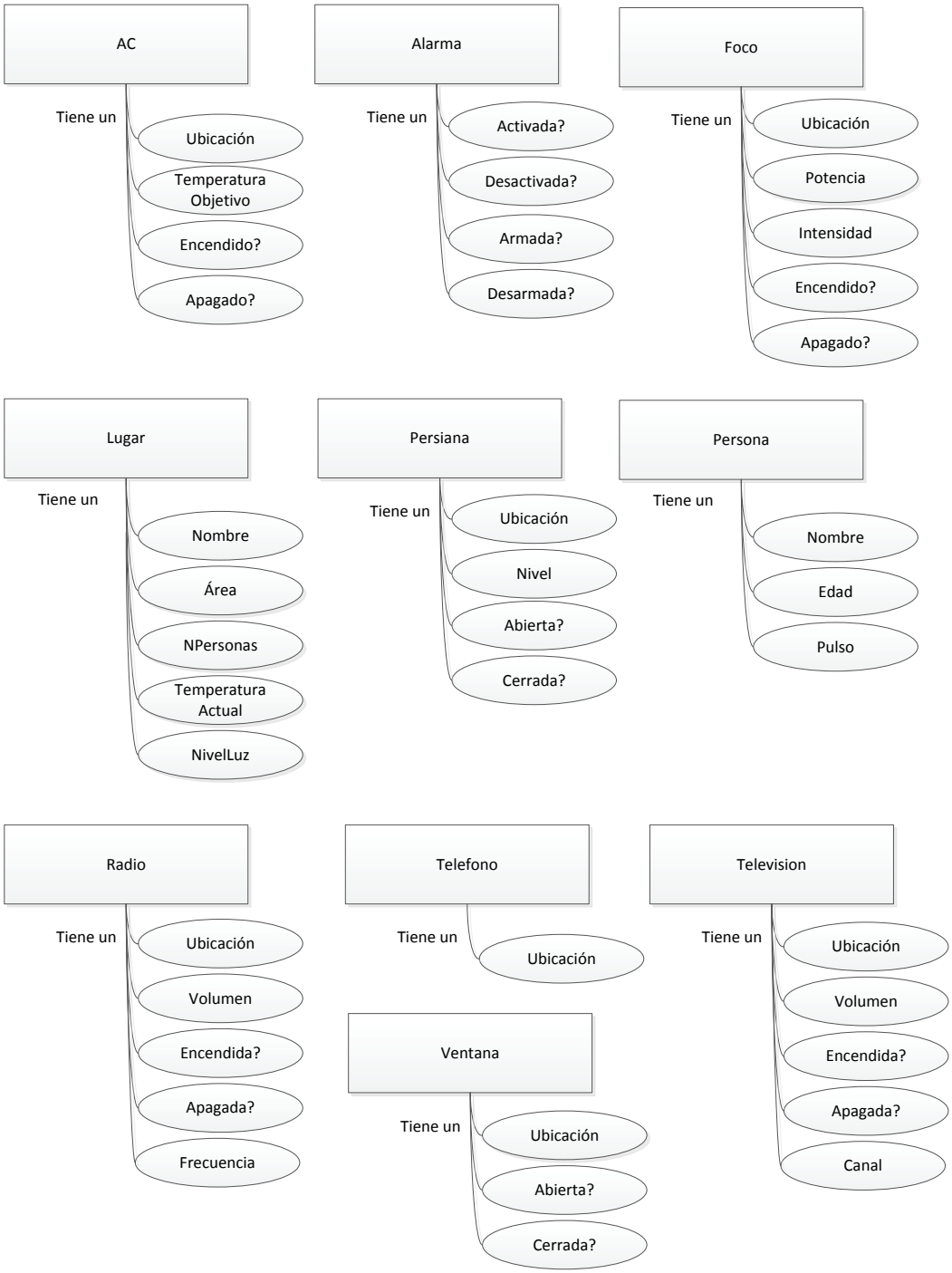
Weiser, M., The computer for the 21st century. *Scientific American*, 265(3), pp.94–104, 1991.

Zhang, T. & Brügge, B., Empowering the user to build smart home applications. En *Proceedings of 2nd International Conference on Smart Homes and Health Telematics (ICOST2004)*. 2004.

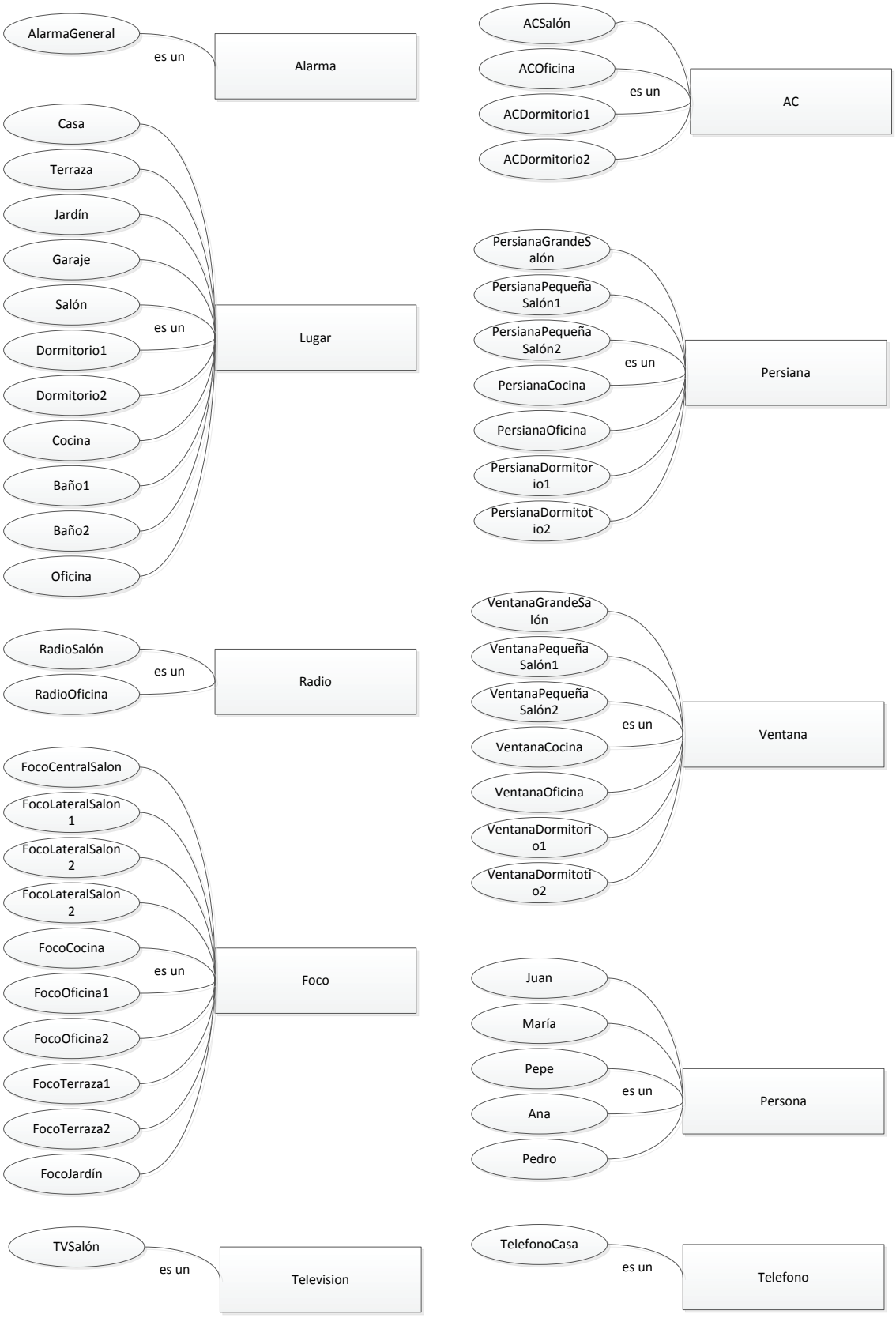
Zorrilla, A.M., *Técnicas de conexión de nodos en mesas interactivas: diseño y evaluacion*. Trabajo Fin de Máster, Departamento de Sistemas Informáticos y Computación, Universidad Politecnica de Valencia. 2013.

Anexo A. Ontología para una casa inteligente

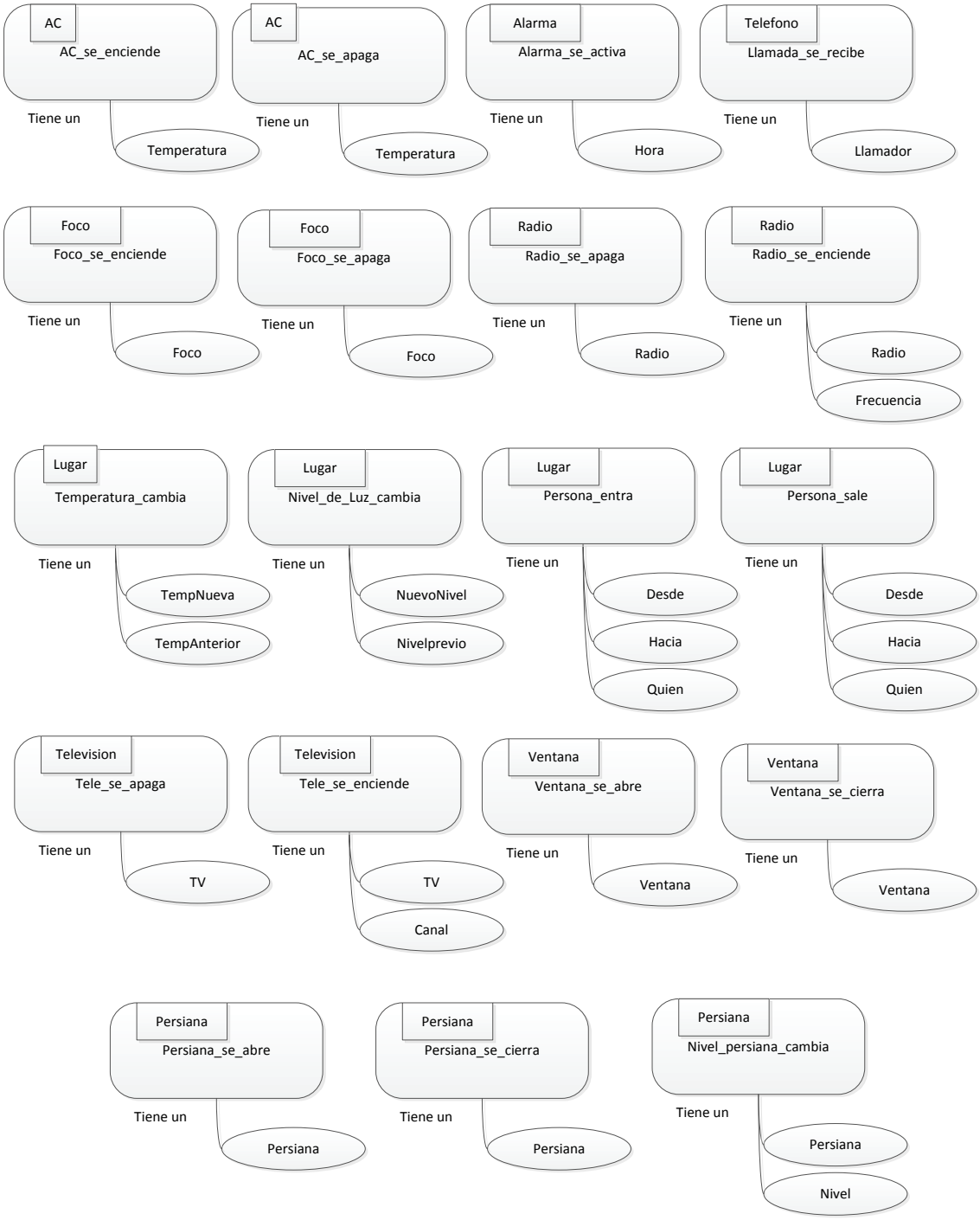
Categorías



Elementos


AlarmaGeneral
es un
Alarma
ACSalón
ACOficina
ACDormitorio1
ACDormitorio2
es un
AC
Casa
Terraza
Jardín
Garaje
Salón
Dormitorio1
Dormitorio2
Cocina
Baño1
Baño2
Oficina
es un
Lugar
PersianaGrandeSalón
PersianaPequeñaSalón1
PersianaPequeñaSalón2
PersianaCocina
PersianaOficina
PersianaDormitorio1
PersianaDormitorio2
es un
Persiana
RadioSalón
RadioOficina
es un
Radio
VentanaGrandeSalón
VentanaPequeñaSalón1
VentanaPequeñaSalón2
VentanaCocina
VentanaOficina
VentanaDormitorio1
VentanaDormitorio2
es un
Ventana
FocoCentralSalon
FocoLateralSalon1
FocoLateralSalon2
FocoLateralSalon2
FocoCocina
FocoOficina1
FocoOficina2
FocoTerraza1
FocoTerraza2
FocoJardín
es un
Foco
Juan
María
Pepe
Ana
Pedro
es un
Persona
TVSalón
es un
Television
TelefonoCasa
es un
Telefono

Sucesos


AC
AC_se_enciende
Tiene un
Temperatura
AC
AC_se_apaga
Tiene un
Temperatura
Alarma
Alarma_se_activa
Tiene un
Hora
Telefono
Llamada_se_recibe
Tiene un
Llamador
Foco
Foco_se_enciende
Tiene un
Foco
Foco
Foco_se_apaga
Tiene un
Foco
Radio
Radio_se_apaga
Tiene un
Radio
Radio
Radio_se_enciende
Tiene un
Radio
Frecuencia
Lugar
Temperatura_cambia
Tiene un
TempNueva
TempAnterior
Lugar
Nivel_de_Luz_cambia
Tiene un
NuevoNivel
Nivelprevio
Lugar
Persona_entra
Tiene un
Desde
Hacia
Quien
Lugar
Persona_sale
Tiene un
Desde
Hacia
Quien
Television
Tele_se_apaga
Tiene un
TV
Television
Tele_se_enciende
Tiene un
TV
Canal
Ventana
Ventana_se_abre
Tiene un
Ventana
Ventana
Ventana_se_cierra
Tiene un
Ventana
Persiana
Persiana_se_abre
Tiene un
Persiana
Persiana
Persiana_se_cierra
Tiene un
Persiana
Persiana
Nivel_persiana_cambia
Tiene un
Persiana
Nivel

Anexo B. Ejercicios de autoevaluación

Este anexo muestra algunos ejemplos de los ejercicios de autoevaluación realizados por los participantes del experimento, la plantilla para la definición de reglas empleada, y ejemplos de ejercicios de escritura y lectura de reglas completas.

Categorías, elementos y propiedades

- Indique cuáles de estos nombres se refieren a categorías con más de dos elementos:
  Televisión PersianaGrandeSalón Salón Persona
- Indique cuáles de estos nombres se refieren a propiedades de Persona:
  Ubicación NivelActual Edad Abierta? Pulso NivelLuz

Sucesos

- Indique cuáles de estos nombres son sucesos originados por la categoría Lugar:
  TemperaturaNueva Entrar Ventana_se_cierra Radio
  Nivel_de_Luz_cambia Temperatura_cambia

Condiciones

Reescriba las siguientes condiciones:

Sentencia	Términos
La persiana de la cocina tiene un nivel de apertura mayor o igual a 10	PersianaCocina.Nivel >=10
Si ocurriera el suceso Temperatura_cambia, cómo expresarías la condición: la nueva temperatura cambiada es mayor que 25	
La radio del salón está encendida y el volumen de la televisión del salón es mayor que 15	

Operaciones

Reescriba las siguientes operaciones:

Sentencia	Términos
Encender la radio del salón	Sí → RadioSalón.Encendida?
Poner el volumen del televisor del salón	

5 unidades por encima del volumen de la radio del salón
Poner la intensidad del foco central del salón a la mitad del nivel de luz del salón

Flujos de datos

Reescriba de forma gráfica las siguientes expresiones:

- Salón.TemperaturaActual + 5 → ACSalón.TemperaturaObjetivo
- Cierto → AlarmaGeneral.Armada?
- Cocina.TemperaturaActual > 28 Y FocoCocina.Intensidad > (20 + FocoCocina.Potencia) → resultado

Plantilla para una regla

Suceso

Origen

Condición

resultado

Destinatario

Operación

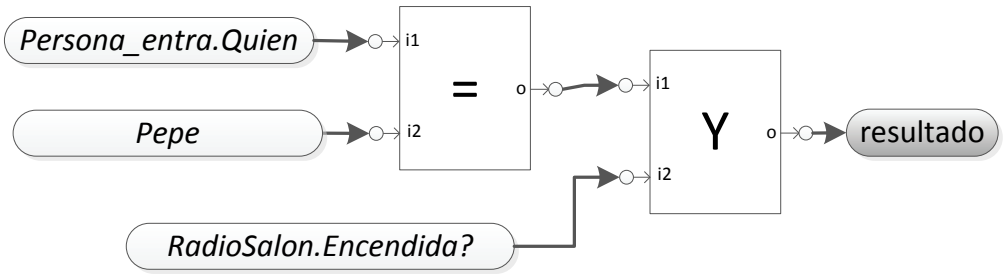
Lectura de reglas

Dada la siguiente regla expresada mediante flujos de datos, escribir la regla equivalente en lenguaje natural.

Suceso Persona_entra

Origen Oficina

Condición



Destinatario RadioOficina

Operación

Escritura de reglas

Dada la siguiente regla escrita en lenguaje natural, rellenar la plantilla correspondiente empleando flujos de datos: “Cuando el salón detecta que la temperatura del salón cambia, y si el nuevo valor de la temperatura es menor que 19 y el numero de personas en el salón es mayor que 2, entonces establecer a 22 grados la temperatura objetivo del aire acondicionado del salón.”

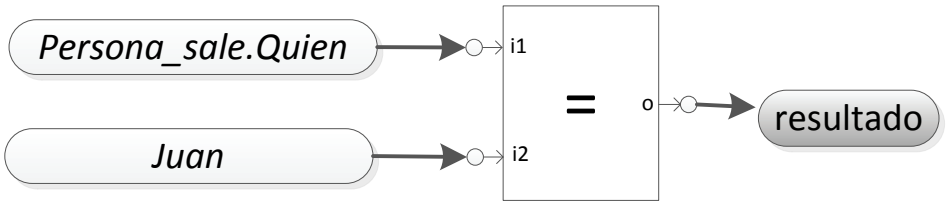
Anexo C. Ejercicios de edición con la herramienta

Tarea 1

Suceso Persona_sale

Origen Casa

Condición



Destinatario AlarmaGeneral

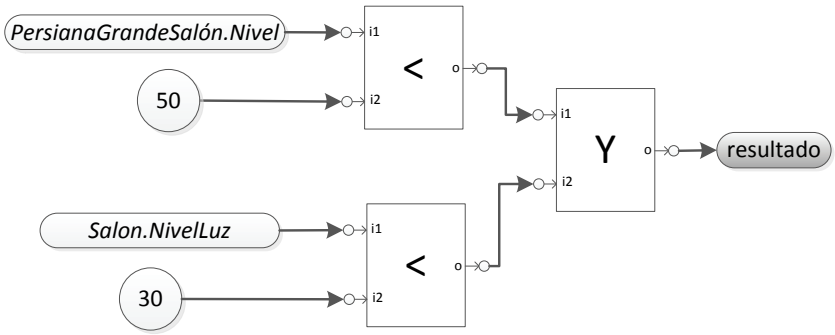
Operación

Tarea 8

Suceso Persiana_se_abre

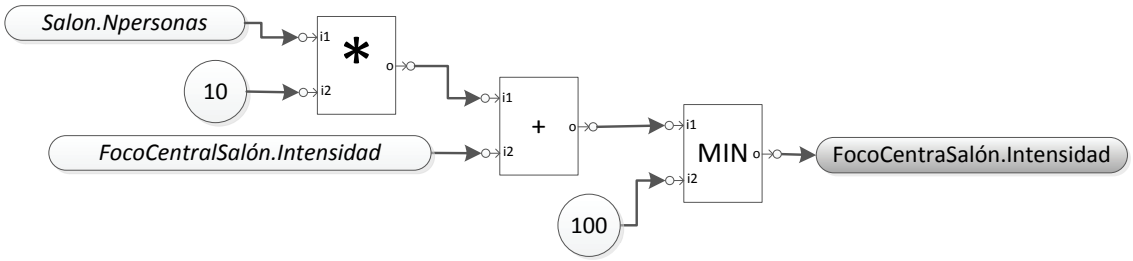
Origen PersianaGrandeSalon

Condición



Destinatario FocoCentralSalon

Operación



Anexo D. Cuestionarios de evaluación de la herramienta

Preguntas de respuesta cerrada

		Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en	De acuerdo	Totalmente de acuerdo
1	Es fácil seleccionar el origen/suceso/destinatario de la regla.					
2	Es fácil añadir operadores nuevos al proceso de datos.					
3	Es fácil añadir elementos ajenos al proceso de datos.					
4	Es fácil añadir propiedades del origen y el destinatario al proceso de datos.					
5	Es sencillo establecer el valor de una constante numérica con el control en forma de regla utilizado.					
6	Conectar dos elementos con un flujo de datos es sencillo e intuitivo.					
7	Crear flujos de datos empleando los dedos resulta una tarea cómoda incluso teniendo que realizar numerosas conexiones.					
8	Es fácil eliminar elementos del proceso de datos.					
9	Resulta útil disponer de tangibles para explorar y posicionar las poblaciones del origen y el destinatario, así como el suceso.					
10	Preferiría realizar la tarea de edición en colaboración con otro usuario.					
11	El editor sería útil si tuviese una mesa interactiva en el contexto de una casa inteligente.					
12	El editor de reglas de comportamiento para superficies interactivas es una herramienta novedosa.					
13	El editor de reglas es fácil de utilizar.					
14	La gente podría aprender a utilizar el editor fácilmente.					
15	No necesitaría importante ayuda técnica para manejar el editor.					